NCAR TECHNICAL NOTES

The Technical Note series provides an outlet for a variety of NCAR manuscripts that contribute in specialized ways to the body of scientific knowledge but which are not suitable for journal, monograph, or book publication. Reports in this series are issued by the NCAR Scientific Divisions. Designation symbols for the series include:

EDD - *Engineering, Design, or Development Reports*
Equipment descriptions, test results, instrumentation, and operating and maintenance manuals.

IA - *Instructional Aids*
Instruction manuals, bibliographies, film supplements, and other research or instructional aids.

PPR - *Program Progress Reports*
Field program reports, interim and working reports, survey reports, and plans for experiments.

PROC - *Proceedings*
Documentation of symposia, colloquia, conferences, workshops, and lectures. (Distribution may be limited to attendees.)

STR - *Scientific and Technical Reports*
Data compilations, theoretical and numerical investigations, and experimental results.

Copyright © 1987, University Corporation for Atmospheric Research. All Rights Reserved.

Published by: National Center for Atmospheric Research
Scientific Computing Division
P.O. Box 3000
Boulder, CO 80307

Additional copies of this manual are available from NCAR Graphics Distribution at the above address. Please contact NCAR Graphics Distribution for price information.

The National Center for Atmospheric Research is operated by the University Corporation for Atmospheric Research and is sponsored by the National Science Foundation. Any opinion, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
## CONTENTS

*Preface* ................................................................................................................................... vii  
*About the Cover* ..................................................................................................................... viii

### Section I: Introduction to NCAR Graphics

1. What Is NCAR Graphics?.................................................................................................................. 3  
2. What Is the Graphical Kernel System Standard?...................................................................................... 3  
3. Overview of Using This Manual......................................................................................................... 4  
4. Overview of the NCAR Graphics Package............................................................................................... 6  
5. Basic Concepts You Need to Understand.............................................................................................. 10  
6. How to Begin Using NCAR Graphics..................................................................................................... 11  
7. Error Messages........................................................................................................................................ 13  
9. History of Graphics at NCAR................................................................................................................ 16  
10. Other Documentation Resources.......................................................................................................... 17  
11. What to Do if You Have Questions or Suggestions........................................................................... 18

### Section II: Guide for Converting Code from the NCAR System Plot Package to the NCAR GKS Graphics Package

1. Introduction........................................................................................................................................... 21  
2. System Plot Package Simulator (SPPS)................................................................................................. 21  
3. The Six Conversion Steps...................................................................................................................... 22  
4. Conversion Example.............................................................................................................................. 30  
5. Former Routines and Utilities Not in NCAR Graphics......................................................................... 34  
6. Appendix A: Notes on FORTRAN 77 TYPE CHARACTER Variables.................................................... 37

### Section III: SPPS — An NCAR System Plot Package Simulator

1. Introduction........................................................................................................................................... 45  
2. Coordinate Systems............................................................................................................................... 46  
3. Coordinate-system Mapping Functions................................................................................................. 47  
4. SPPS Subroutines and Their Relationship to GKS............................................................................... 48  
5. Conversion Considerations................................................................................................................... 56
**Section IV: NCAR Graphics Utilities, GKS Level 0A**

- Introduction ........................................................................................................... 63
- Changes from the Previous Version ........................................................................... 63
- Graphics Utilities Categorized by Functionality .................................................. 63
- Brief Descriptions of the Graphics Utilities ......................................................... 64
- Detailed Descriptions of Each Graphics Utility .................................................... 67

- AREAS .................................................................................................................... 69
- AUTOGRAPH ......................................................................................................... 89
- CONRAN ............................................................................................................... 105
- CONRAQ .................................................................................................................. 123
- CONRAS .................................................................................................................. 137
- CONREC .................................................................................................................. 155
- CONRECQCK ......................................................................................................... 163
- CONRECSPR .......................................................................................................... 169
- DASHCHAR .............................................................................................................. 177
- DASHLINE ............................................................................................................. 183
- DASHSMTH ............................................................................................................ 187
- DASHSUPR ............................................................................................................ 193
- EZMAP ................................................................................................................... 201
- EZMAPA ............................................................................................................... 287
- GRIDAL .................................................................................................................. 303
- HAFTON ................................................................................................................. 313
- HISTGGR ............................................................................................................... 321
- ISOSRFX ................................................................................................................. 333
- ISOSEFHR ............................................................................................................. 339
- PWRITX .................................................................................................................. 345
- PWRITY .................................................................................................................. 353
- PWRZI .................................................................................................................... 357
- PWRZS .................................................................................................................... 361
- PWRZT .................................................................................................................... 365
- SRFACE .................................................................................................................. 369
- STRMLN .................................................................................................................. 377
- THREED .................................................................................................................. 385
- VELVCT .................................................................................................................... 391

**Section V: Plot Examples with Source Codes**

- How This Section Is Organized ............................................................................. 401
- Higher-level Utilities ............................................................................................. 403

- AREAS Example .................................................................................................. 405
- AUTOGRAPH Example ......................................................................................... 413
- Standard Contouring ............................................................................................. 421
  - CONREC + DASHCHAR Example ....................................................................... 421
  - CONRAN + DASHCHAR Example ..................................................................... 426
- Quick Contouring .................................................................................................. 431
  - CONRECQCK + DASHLINE Example ............................................................... 431
  - CONRAQ + DASHLINE Example ..................................................................... 435
PREFACE


The authors want to acknowledge Dave Robertson and Tom Wright for their work in creating and documenting the original versions of many of the utilities. Thanks also to all the programmers and users who have tested and refined the utilities and documentation in the years since the first NCAR graphics package was introduced.

The authors wish to thank Nancy Dawson for editing and rewriting portions of this manual and Sandy McCullough for her attention to details in indexing, proofreading, and preparing final copy for printing. Thanks to Christine Guzy for extensive word processing and formatting. Thanks to Paul Bailey, Bob Chatfield, Bob Chervin, Don Cahoon, Lester Lamhut, Larry Lyjak, Andy Mai, Roy Mendelson, Karl Sierka, and Dick Valent for their documentation suggestions, many of which have been incorporated here. Thanks also for the suggestions and comments from many other users of the previous NCAR graphics package and documentation.

The authors also wish to recognize other contributors to this manual: Belinda Housewright and Lisa Benson for word processing; Bob Nicol for computer formatting assistance; Diane Huntrods for proofreading; Paul Krueger for work on HISTGR and the cover; Ginger Caldwell and Dee Copelan for their management of the documentation effort; Barbara Mericle for cover graphics work; and Bob Bumpas for the cover photography.
ABOUT THE COVER

The color globes on the cover were created by a program that uses the utilities AREAS, EZMAP, and EZMAPA, which were written by Dave Kennison. To produce the plots, a metafile was translated and displayed on a TEKTRONIX® 4115 terminal. The plots were photographed from the screen, using a large format camera (Hasselblad) and Kodak Ektachrome ASA 200 film. The code for the cover plots is in the "Examples" section.

®TEKTRONIX is a registered trademark of Tektronix, Inc.
Section I: Introduction to NCAR Graphics


What Is the Graphical Kernel System Standard? ............................. 3

Overview of Using This Manual .................................................. 4
  How This Manual Is Organized .................................................. 4
  How This Manual Differs from the Last One ............................... 5
  How to Use This Manual ....................................................... 5

Overview of the NCAR Graphics Package .................................. 6
  Strengths of the NCAR Graphics Package ................................ 6
  The Code for the NCAR Graphics Package ............................... 6
  Brief Descriptions of the NCAR Graphics Utilities .................... 7
  Relationship of Package Components ....................................... 9

Basic Concepts You Need to Understand .................................... 10
  Graphical Kernel System (GKS) ............................................. 10
  Device Independent Graphics ................................................. 10
  Metafile and Metacode ........................................................... 10
  Computer Graphics Metafile (CGM) ......................................... 10
  NCAR System Plot Package Simulator (SPPS) ........................... 11
  Higher-level Utilities ............................................................. 11
  Lower-level Subroutines ....................................................... 11
  NCAR Graphics ................................................................. 11
  NCARGRAPHICS ................................................................. 11

How to Begin Using NCAR Graphics ......................................... 11
  What to Expect from the GKS Level You Are Using .................. 11
  How Much Detail You Need to Know About the Standards ......... 12
  GKS Calls You Must Know ..................................................... 12
    Opening GKS and Opening and Activating a Workstation ....... 12
    Deactivating and Closing the Workstation and Closing GKS .. 13

Error Messages ........................................................................ 13

Status of Color Interface for NCAR Graphics ............................ 14

History of Graphics at NCAR .................................................. 16

Other Documentation Resources .............................................. 17
  NCAR Graphics Installation ................................................. 17
  GKS .................................................................................... 17
  CGM .................................................................................... 17

What to Do if You Have Questions or Suggestions ....................... 18
Introduction to NCAR Graphics

What Is NCAR Graphics?

NCAR Graphics is a collection of FORTRAN 77 programs and subroutines that can be used to generate and plot computer graphics suitable for the display of scientific data. NCAR Graphics conforms to the Graphical Kernel System (GKS) standard, Level 0A (zero A).

Individual NCAR Graphics utilities
• produce X-Y coordinate plots.
• contour data fields on regularly-spaced and irregularly-spaced grids.
• produce world maps using one of ten projections.
• produce solid-colored maps.
• display two-dimensional vector fields.
• draw lines in three-dimensional space.
• provide halftone (gray scale) capabilities.
• produce background grids for graph paper.
• draw three-dimensional displays of functions of two variables.
• draw iso-surfaces from a three-dimensional array.
• draw text in various fonts.
• draw dashed lines with user-defined patterns.

Some of the utilities can be used together. For example, contour plots can be superimposed on world map projections.

What Is the Graphical Kernel System Standard?

GKS is a set of basic functions for computer graphics programming; it can be used by many graphics-producing applications. The GKS standard has been adopted by both the American National Standards Institute (ANSI) and the International Standards Organization (ISO). GKS can be implemented in one of twelve levels, depending on the graphical input and output capabilities. The GKS standard also specifies how the defined graphics functionality must be implemented in FORTRAN. NCAR Graphics contains a full FORTRAN 77 implementation of GKS at Level 0A. GKS at Level 0A has roughly the same functionality as the System Plot Package in previous NCAR graphics packages.
Overview of Using This Manual

This manual is a reference guide for users of the NCAR Graphics package. The manual is for use by two audiences: people who have never used a previous NCAR graphics package, and those who have used previous NCAR graphics packages and now need to convert to NCAR Graphics. Both groups need to read this "Introduction" section of the manual to get an overview of how NCAR Graphics works and to determine which other parts of the manual to use.

How This Manual Is Organized

The major tabs divide this manual into five sections:

Introduction: This is the section you are now reading. It describes what the manual contains and how to use the manual. The section also defines some key concepts, gives an overview of NCAR Graphics, and reports on the status of the color interface.

Conversion Guide: The full name is "Guide for Converting Code from the NCAR System Plot Package to the NCAR GKS Graphics Package." Considerable effort was expended to make NCAR Graphics compatible with previous NCAR graphics packages. However, the conversion from FORTRAN 66 to FORTRAN 77 required some changes. If you have applications that are dependent on the NCAR System Plot Package (NSPP), use the "Conversion Guide" to make those applications run under the new NCAR Graphics system.

If you do not have any files that depend on previous NCAR graphics packages, you can skip the Conversion Guide section entirely.

SPPS: The full name is "SPPS — An NCAR System Plot Package Simulator." The support package for pre-GKS NCAR Graphics was the NCAR System Plot Package (NSPP). In the NCAR GKS package, NSPP has been replaced by GKS. SPPS was written to simulate the old NSPP package in the GKS environment, but there are some differences between the two. This manual section describes the differences and explains the coordinates for SPPS subroutines. If you have graphics applications that depend on NSPP, you will need to use the SPPS section of the manual.

SPPS offers a user interface to GKS that is independent of the use of SPPS as a conversion aid. You may find it easier to use SPPS entries than to call GKS entries directly.

Graphics Utilities: The detailed descriptions of the higher-level graphics utilities and their options and argument lists are in this section. The utilities are arranged in alphabetical order. This section contains output examples for EZMAP and the FORTRAN source code for those examples. For examples of other utilities, see the "Examples" section.

Examples: This section contains the FORTRAN code and output for examples of each utility. It also contains examples for SPPS entries, complex
plots, and GKS output primitives. Ask your NCAR Graphics site representa-
tive where to find the on-line codes that generate these examples so that
you can use them as starting points for producing your own applications.

How This Manual Differs from the Last One

The previous NCAR GKS manual has been reorganized into two manuals.
The manual you are now reading, *NCAR Graphics User's Guide*, concen-
trates on what users need to know to use NCAR Graphics. *NCAR Graphics
Installer's Guide* (NCAR/TN-284+IA) contains the information needed by
the person who installs and maintains NCAR Graphics at your location.
These two manuals replace *The NCAR GKS-Compatible Graphics System
* (NCAR/TN-267+IA), April 1986. The new manuals contain several
significant changes. In the manual you are now reading,

- The "Introduction" and "Conversion Guide" sections have been rewritten
to include additional material. Minor changes have been made to the
"SPPS" section.
- The new utilities AREAS and EZMAPA have been added.
- The former utility HSTGMR has been rewritten to correct errors and
significant new features have been added. Because of these changes, the
old name was dropped and the new utility is called HISTGR.
- A shortened description of AUTOGRAPH has been added to the "Utili-
ties" section. The AUTOGRAPH example codes were rewritten for exe-
cution in a GKS environment. The unabridged AUTOGRAPH documen-
tation, *AUTOGRAPH: A Graphing Utility*, (NCAR/TN-245+IA), has been
updated and is available as a separate manual, which was shipped to you
along with this manual.
- Two changes were made in EZMAP. MAPEOS was replaced with
MAPEOD, which has different arguments, and the seven intensity param-
eters were replaced by color-index parameters. The EZMAP example
codes were rewritten for execution in a GKS environment.
- All of the utilities have been updated to incorporate minor changes.
- Many more examples have been added in the "Examples" section.
- This manual has an index, main table of contents, section table of con-
tents, and consecutive page numbering.

How to Use This Manual

To find what you are looking for, use the main table of contents at the
beginning of the manual, the more detailed table of contents at the begin-
ing of each section, or the index.

Most users will not need to use all parts of this manual. As you use it, you
may want to add more tabs to help locate the sections you use frequently.
We recommend that you put the manual in a 3-inch, D-ring binder, which is
available from office supply stores. The rings are shaped like the letter "D,"
which allows pages to turn easily and helps the pages lie flat when the book
is open.
Overview of the NCAR Graphics Package

Strengths of the NCAR Graphics Package

The NCAR Graphics package has these strengths:

- **Functionality:** The package offers a wide range of capabilities for the display of scientific data.

- **Portability:** The package conforms to three national standards: the FORTRAN language standard; the Graphical Kernel System (GKS) graphics standard; and the Computer Graphics Metafile (CGM) standard for metafiles. Conformance to national standards allows codes to be ported to any computing environment that supports those standards. Conforming to national standards also provides a common basis for understanding.

- **Reliability:** The GKS standard is the result of dozens of full-time equivalent staff years of effort. The GKS standard represents an effort to specify a consistent set of graphical functions that can be used by the majority of applications that produce computer-generated pictures. The standard solves graphics problems in an organized and non-conflicting manner.

- **Potential Growth:** Utilizing the GKS standard provides a solid basis for future expansion in the areas of color representation, picture segmentation, raster display, and interactive input. Also, any GKS-based products produced independently from NCAR can be integrated into the package with minimal effort.

- **Source Availability:** The source code for the package is provided, which allows for user-modification to meet specific needs. If a code does not do exactly what you want, you can frequently insert modifications to produce the desired effects. The source code is copyrighted by UCAR and redistribution is permitted only by arrangement with UCAR.

The Code for the NCAR Graphics Package

Your NCAR Graphics site representative should make the following codes available to you:

- **A GKS package, level 0A at minimum:** This may be a commercial product, or it may be the GKS package written by NCAR. The NCAR code allows for only one active workstation: a Metafile Output (MO) workstation. The format of the metafile created adheres to the CGM standard.

- **SPPS, the System Plot Package Simulator:** This package consists of GKS-based routines that simulate most entries in the old NCAR System Plot Package. This system will greatly assist users of the old package who want to convert to GKS and will provide some useful entry points to new users. You should read the "SPPS" section carefully for details.

- **Higher-level utilities:** These are the primary user interface to the package. FORTRAN utilities are provided for contouring, world map projections, automatic graphing, three-dimensional line drawing, fancy
Introduction

characters, flow-field display, halftone (gray-scale) plotting, and so on. A brief overview of all the utilities is presented here and also at the beginning of the "Utilities" section. You may want to thumb through the "Examples" section of this manual to get a feel for what the package can produce.

- **User aids:** The routine FINDG will aid in converting old program units to the new package; it will locate all occurrences of calls to old utilities and calls to NCAR System Plot Package entries. FINDG is described in the "Conversion Guide" section of this manual.

- **Demonstration drivers:** This is a collection of simple test drivers for all higher-level utilities.

- **Metafile translator.** Your NCAR Graphics site representative should have implemented the NCAR CGM Translator so that a simple invocation will produce plots on the desired output device.

**Brief Descriptions of the NCAR Graphics Utilities**

Here is a list of the higher-level utilities and a brief description of what each one does. The utilities are listed here by the 8-character-or-less name that is used on the NCAR Graphics distribution tape. Three of the utilities are most commonly referred to by longer names — AUTOGRAPH (for AUTOGRPH), CONRECQCK (for CONRCQCK), and CONRECSPR (for CONRCSPR). For the details of using each utility, turn to the "Utilities" section of this manual. You may want to leaf through the "Examples" section in the back of the book to get an idea of the output possibilities.

**AREAS:** Creates an area map from a set of edges that divide a two-dimensional plane into areas. The area map may then be used in a number of ways, such as in conjunction with EZMAPA to create solid-colored maps.

**AUTOGRPH:** Draws and annotates curves or families of curves. This utility is also referred to as AUTOGRAPH.

**CONRAN:** Contours irregularly spaced data, labeling the contour lines.

**CONRAQ:** Like CONRAN, but smaller and faster because it has no labeling capacity.

**CONRAS:** Like CONRAN, but bigger and slower because lines are smoothed and crowded lines are removed.

**CONREC:** Contours two-dimensional arrays, labeling the contour lines.

**CONRCQCK:** Like CONREC, but faster and smaller because contours are unlabeled. This utility is also referred to as CONRECQCK.

**CONRCSPR:** Like CONREC, but bigger and slower because contours are smoothed and labeled, and crowded lines are removed. This utility is also referred to as CONRCSPR.

**DASHCHAR:** Software dashed-line package with labeling capability.
**DASHLINE**: Like DASHCHAR, but smaller and faster because it has no labeling capability.

**DASHSMTH**: Like DASHCHAR, but bigger and slower because lines are smoothed.

**DASHSUPR**: Like DASHCHAR, but bigger and slower because lines are smoothed and crowded lines are removed.

**EZMAP**: Plots continental, U.S. state, and/or world political outlines according to one of ten projections.

**EZMAPA**: Allows EZMAP output to be redirected to routines in the AREAS package. EZMAPA makes it possible to create solid-colored world maps and to draw lines on a map, masked by the areas created by the area map (lines of latitude and longitude omitted over land masses, for example).

**GRIDAL**: Package for drawing graph paper, backgrounds, perimeters, and so on.

**HAFTON**: Halftone (gray scale) pictures from a two-dimensional array.

**HISTGR**: General purpose package for drawing histograms (bar charts).

**ISOSRF**: Iso-surfaces (with hidden lines removed) from a three-dimensional array. May optionally include PWRZI for drawing characters in three space.

**ISOSRFHR**: Iso-surfaces (with hidden lines removed) from a high resolution three-dimensional array.

**PWRITX**: Draws fancy characters, using the Hershey database.

**PWRITY**: Draws simple software characters.

**PWRZI**: To be used with ISOSRF to draw characters in three space.

**PWRZS**: To be used with SRFACE to draw characters in three space.

**PWRZT**: To be used with THREED to draw characters in three space.

**SRFACE**: Three-dimensional display of a function of two variables (with hidden lines removed). May optionally include PWRZS, for drawing characters in three space.

**STRMLN**: Plots a representation of any two-dimensional vector field for which planar vector components are given on a regular rectangular lattice, displaying both field direction (via lines of flow containing arrowheads and feathers) and field magnitude (based on distance between those flow lines).

**THREED**: Provides three-space line drawing capabilities, with entry points equivalent to the line drawing entry points of the System Plot Package
Simulator. May optionally include PWRZT for drawing characters in three space.

VELVCT: Two-dimensional velocity field displayed by drawing arrows from the data locations.

**Relationship of Package Components**

It is important to understand how the components of the NCAR GKS Graphics package relate to one another in terms of the call tree and logic flow. The following diagram presents an overview:

**Figure 1 — Components of NCAR Graphics**

In the top box of the diagram, a component sitting on top of another one indicates that program units in the upper component can call any of the components in the lower packages. For example, application programs can
call GKS functions directly, or applications programs can call the utilities or SPPS entries. Likewise, the utilities can make direct GKS calls or calls to SPPS entries.

The execution of the NCAR GKS package generates a Computer Graphics Metafile, which can be used as input to the NCAR CGM Translator and subsequently displayed on a graphics output device.

Note that this diagram pertains to the NCAR GKS package only, which is at GKS Level 0A. If you are using a higher level GKS package, there is also a direct path from GKS to the graphics output device, as well as a path for graphical input.

The insertion of the metafile between an application program and graphics output devices greatly simplifies the problem of running applications on many different host computers and producing graphics output on many different output devices. Metafiles can also be used for archives of graphics instructions and for transporting graphics instructions to other computers.

Basic Concepts You Need to Understand

Graphical Kernel System (GKS)

GKS is a set of basic functions for computer graphics programming; it can be used by many graphics-producing applications. The GKS standard has been adopted by both the American National Standards Institute (ANSI) and the International Standards Organization (ISO). GKS can be implemented in one of twelve levels, depending on the graphical input and output capabilities. The GKS standard also specifies how the defined graphics functionality must be implemented in FORTRAN. NCAR Graphics contains a full FORTRAN 77 implementation of GKS at Level 0A. GKS at Level 0A has roughly the same functionality as the System Plot Package in previous NCAR graphics packages.

Device Independent Graphics

Graphics files that can be executed on any host computer and can produce plots on any graphics output device are called device independent graphics. Adopting the GKS standard is one step toward making graphics device independent.

Metafile and Metacode

A metafile is a file of encoded graphics instructions that are not specific to any particular output device. Metafiles are sometimes also referred to as metacode.

Computer Graphics Metafile (CGM)

The Computer Graphics Metafile (CGM) is an ANSI and ISO standard for defining metafiles. The NCAR GKS package produces a CGM.
NCAR System Plot Package Simulator (SPPS)

SPPS is a collection of low-level graphics subroutines. It is an interface that calls GKS routines. SPPS was written to emulate the NCAR System Plot Package (NSPP), which was part of previous NCAR graphics packages.

Higher-level Utilities

All the NCAR utilities that are described in the "Utilities" section of this manual are referred to as higher-level utilities. They have "higher-level" capabilities. CONRAN, for example, draws contours of data defined on non-equally spaced grids.

Lower-level Subroutines

The terms lower-level subroutines or lower-level utilities are used loosely to mean any entry in NCAR Graphics that is not a higher-level utility. The SPPS entry LINE is an example.

NCAR Graphics

The term NCAR Graphics refers to the totality of the NCAR GKS-based graphics package. NCAR Graphics is at GKS level 0A.

NCARGRAPHICS

The term NCARGRAPHICS (with this spelling) is used to refer to everything that is on the distribution tape.

How to Begin Using NCAR Graphics

To create plots on an output device, load and execute the following:

- your application program.
- all necessary higher-level NCAR graphics utilities.
- SPPS.
- locally-implemented support subroutines.
- a GKS package of at least Level 0A.

Your NCAR Graphics site representative may have made this easy for you by creating a program library containing the higher-level graphics utilities, SPPS, and the locally-implemented support subroutines.

What to Expect From the GKS Level You Are Using

The NCAR-supplied GKS package is at Level 0A. The only possible output is a metafile. To produce plots from the metafile on a graphics output device, it is necessary to invoke the metafile translator. For information on how to invoke the translator at your site, see your NCAR Graphics site representative.
Many people use NCAR Graphics with a commercial GKS package implemented at a higher level than 0A. If your GKS package is at a higher level than 0A, the executing program may generate a metafile, produce plots directly on the output device, and allow for graphical input. Ask your NCAR Graphics site representative about the capabilities of the GKS level installed at your site.

How Much Detail You Need to Know About the Standards

NCAR Graphics conforms to three ANSI standards: FORTRAN 77, GKS, and Computer Graphics Metafile (CGM). It is important to note that it is quite possible to use the NCAR Graphics package with little or no knowledge of any of the ANSI standards upon which it is based. The primary user interface is FORTRAN subroutine calls to the higher-level graphics utilities.

However, if you want to write portable FORTRAN code, then you must understand the FORTRAN standard. If you want to use features of GKS directly, then it is necessary to understand that standard. It is not necessary for the average user to understand the CGM standard, since your NCAR Graphics site representative's implementation of the NCAR CGM Translator should make the translation of the metafile automatic.

GKS Calls You Must Know

The only GKS calls that you need to know are the ones to open and close GKS and the ones to open, activate, deactivate, and close a workstation. You must open GKS and open and activate a workstation before calling any of the higher-level graphics utilities.

Opening GKS and Opening and Activating a Workstation: All GKS calls begin with the letter G. To open GKS, issue the GKS call:

```fortran
CALL GOPKS (IER, ISZ)
```

IER (usually the standard output unit 6) is the logical unit to receive GKS error messages. ISZ is the maximum buffer size allowed by GKS. In the NCAR GKS package, ISZ is a dummy argument, but it must appear in the argument list. If you are using another GKS package, consult your NCAR Graphics site representative to determine what ISZ should be.

To open a workstation, issue the call:

```fortran
CALL GOPWK (1, 2, 1)
```

This call establishes the workstation identifier as 1, the FORTRAN logical unit number for output as 2, and the workstation type as 1 (the value for the type of the Metafile Output workstation in the NCAR GKS 0A package). With the routine G01MIO that is provided in the NCAR GKS package, the invocation GOPWK will open FORTRAN logical unit 2 with file name GMETA.
Introduction

To activate the workstation, issue the call:

CALL GACWK (1)

If you are using NCAR GKS 0A, you can use

CALL OPNGKS

in place of the three calls described above. OPNGKS is part of SPPS.

Deactivating and Closing the Workstation and Closing GKS: When you are finished using NCAR Graphics, the workstation must be deactivated and closed and GKS must be closed. To deactivate and close the workstation, issue the calls:

CALL GDAWK (1)
CALL GCLWK (1)

where 1 is the workstation identifier. To close GKS, issue the call:

CALL GCLKS

If you are using NCAR GKS 0A, you can use

CALL CLSGKS

in place of the three calls described above. CLSGKS is part of SPPS.

Error Messages

The GKS standard identifies specific error conditions and associates a specific error number with each identified error condition. All GKS implementations use the same error numbers, but packages may differ as to the format of how the error messages are issued. In the NCAR GKS package, the subroutine in which the error occurred is identified in the first line of the error message. The error message starts with "GKS...." For example:

GKS ERROR NUMBER 5 ISSUED FROM SUBROUTINE GPL:
—GKS NOT IN PROPER STATE: GKS SHALL BE EITHER IN THE STATE WSAC OR IN THE STATE SGOP

Error messages from the NCAR graphics utilities are numbered and the subroutine is identified in the first line. The error message starts with "ERROR...." For example:

```
ERROR 1 IN CONREC - DIMENSION ERROR - M*N .GT. (2**IARTH) M = 0  N = 25
```

Status of Color Interface for NCAR Graphics

A distinct advantage of GKS is that it provides a well-defined color interface. However, the color interface has not been completely implemented and tested in the NCAR GKS 0A package as of this writing. Information on the current NCAR Graphics color interface follows.

The default line color and text color is the foreground color in NCAR Graphics. A crude interface has been implemented for changing major and minor line colors and text color in the output from the utilities. AUTOGRAPH, EZMAP, EZMAPA, and HISTGR have their own color access interface, so you should consult the documentation in the "Utilities" section to learn the interface for those utilities.

All other utilities have a common block that contains the color indices to be used for major lines, minor lines, and text. The common blocks and names of the color indices are listed in Table 1.
## Table 1 — NCAR GKS 0A Color Interface

<table>
<thead>
<tr>
<th>Utility name</th>
<th>Common block</th>
<th>Major lines</th>
<th>Minor lines</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONRAN</td>
<td>RANINT</td>
<td>IRANMJ</td>
<td>IRANMN</td>
<td>IRANTX</td>
</tr>
<tr>
<td>CONRAQ</td>
<td>RAQINT</td>
<td>IRAQMJ</td>
<td>IRAQMN</td>
<td>IRAQTX</td>
</tr>
<tr>
<td>CONRAS</td>
<td>RASINT</td>
<td>IRASMJ</td>
<td>IRASMN</td>
<td>IRASTX</td>
</tr>
<tr>
<td>CONRECQCK</td>
<td>QCKINT</td>
<td>IQCKMJ</td>
<td>IQCKMN</td>
<td>IQCKTX</td>
</tr>
<tr>
<td>CONRECSPR</td>
<td>SPRINT</td>
<td>ISPRMJ</td>
<td>ISPRMN</td>
<td>ISPRTX</td>
</tr>
<tr>
<td>CONREC</td>
<td>RECINT</td>
<td>IRECMJ</td>
<td>IRECMN</td>
<td>IRECTX</td>
</tr>
<tr>
<td>DASHCHAR</td>
<td>DCHINT</td>
<td>IDCCHMJ</td>
<td>IDCCHMN</td>
<td>IDCHTX</td>
</tr>
<tr>
<td>DASHLINE</td>
<td>DLNINT</td>
<td>IDLNMJ</td>
<td>IDLNMN</td>
<td>IDLNTAX</td>
</tr>
<tr>
<td>DASHSMTH</td>
<td>DSMINT</td>
<td>IDSMMJ</td>
<td>IDSMMN</td>
<td>IDSMTX</td>
</tr>
<tr>
<td>DASHSUPR</td>
<td>DSPINT</td>
<td>IDSPMJ</td>
<td>IDSPMN</td>
<td>IDSPTX</td>
</tr>
<tr>
<td>GRIDAL</td>
<td>GRIINT</td>
<td>IGRIMJ</td>
<td>IGRIMN</td>
<td>IGRTX</td>
</tr>
<tr>
<td>HAFTON</td>
<td>HAFINT</td>
<td>IHAFMI</td>
<td>IHAFMN</td>
<td>IHAFTX</td>
</tr>
<tr>
<td>ISOSRF</td>
<td>ISOINT</td>
<td>IIOSMJ</td>
<td>IIOSMN</td>
<td>IIOTX</td>
</tr>
<tr>
<td>ISOSRFHR</td>
<td>IHRINT</td>
<td>IIHRMJ</td>
<td>IIHRMN</td>
<td>IIHRTX</td>
</tr>
<tr>
<td>PWRITX</td>
<td>PWXINT</td>
<td>IPWXMJ</td>
<td>IPWXMN</td>
<td>IPWXTX</td>
</tr>
<tr>
<td>PWRITY</td>
<td>PWINT</td>
<td>IPWYMJ</td>
<td>IPWYMN</td>
<td>IPWTX</td>
</tr>
<tr>
<td>PWRZI</td>
<td>PWZIINT</td>
<td>IPZIMJ</td>
<td>IPZIMN</td>
<td>IPZITX</td>
</tr>
<tr>
<td>PWRZS</td>
<td>PZSINT</td>
<td>IPZSMJ</td>
<td>IPZSMN</td>
<td>IPZSTX</td>
</tr>
<tr>
<td>PWRZT</td>
<td>PZTINT</td>
<td>IPZTMJ</td>
<td>IPZTMN</td>
<td>IPZTXT</td>
</tr>
<tr>
<td>SRFACE</td>
<td>SRFINT</td>
<td>ISRFMJ</td>
<td>ISRFMN</td>
<td>ISRFTX</td>
</tr>
<tr>
<td>STRMLN</td>
<td>STRINT</td>
<td>ISTRMJ</td>
<td>ISTRMN</td>
<td>ISTRTX</td>
</tr>
<tr>
<td>THREED</td>
<td>THRINT</td>
<td>ITHRMJ</td>
<td>ITHRMN</td>
<td>ITHRXT</td>
</tr>
<tr>
<td>VELVCT</td>
<td>VELINT</td>
<td>IVELMJ</td>
<td>IVELMN</td>
<td>IVELTX</td>
</tr>
</tbody>
</table>

All three color indices are initialized to "1," the foreground color. If you want to change colors, add the appropriate common block and set the appropriate color index for major lines, minor lines, or text with the desired specification for that color index. Note that the colors available vary depending on the type of output device you are using. Some utilities, such as SRFACE, VELVCT, and STRMLN, have no distinction between major and minor lines — in such cases, use only the settings for major lines. Consult the manual for your output device to determine what colors are available. See the following example.
For example, if you want to change the color of the lines to red in `SRFACE`, you could do so by inserting the following into your code:

```
COMMON /SRFINT/ ISRFMJ, ISRFMN, ISRFEX

C
C Set color index 2 on workstation 1 to red.
C
CALL GSCR(1, 2, 0.0, 0.0, 0.0)
C
C Set SRFACE line color to index 2.
C
ISRFMJ = 2

```

The color interface has not been thoroughly tested. If you use it, please report your results to: NCAR/Scientific Computing Division, Distributed Computing Group/Graphics, P.O. Box 3000, Boulder, CO 80307.

**History of Graphics at NCAR**

The NCAR graphics package dates from the mid-1960s. The initial impetus for the package was to produce a set of plotting tools to support NCAR scientists in their research. Most of the code has been written by the staff of the Scientific Computing Division of NCAR in response to requests from, and perceived needs in, the scientific community.

In order to satisfy the needs of a rapidly growing user base, particularly those at remote sites, it became clear at a very early stage that portability and device independence were desirable goals for the package. It is these features that have made the NCAR package successful over the last two decades.

A major step toward device independence was taken in the early 1970s when the package was converted to produce metacode rather than device dependent instructions. In the 1980s, several additional steps have been taken toward standardizing the NCAR graphics library and making it more portable. NCAR Graphics is a result of these standardization efforts. The current NCAR Graphics package conforms to three ANSI standards: FORTRAN 77, GKS (graphics), and CGM (metafile).
Other Documentation Resources

NCAR Graphics Installation

If you need information about the NCAR Computer Graphics Metafile (CGM), the NCAR CGM Translator, GRAPHCAPs and FONTCAPs, the files on the distribution tape, or instructions for installing the NCAR Graphics package, see *NCAR Graphics Installer’s Guide* (NCAR/TN-284+IA). Your NCAR Graphics site representative should have a copy.

GKS

The ultimate source for learning about GKS is the actual ANSI standard: *Computer Graphics — Graphical Kernel System (GKS) Functional Description*. This is standard number ANSI X3.124-1985. The FORTRAN Binding, ANSI X3.124.1-1985, is also contained in this document. The FORTRAN Binding specifies how the GKS functions are to be implemented in FORTRAN; it provides the subroutine names and descriptions of the arguments. A copy of the GKS standard can be obtained from:

ANSI
1430 Broadway
New York, NY 10018
(212) 354-3300

The actual standard is a little dense (as are most standards descriptions). Fortunately, there are several other reference documents on GKS that make it easier to understand:


- *Computer Graphics Programming GKS — The Graphics Standard*, by G. Enderle et. al. (Springer-Verlag, 1984) is excellent and readable. This book provides a complete description of GKS, a description of all GKS error messages, and many examples.

- The August 1986 issue of "IEEE Computer Graphics and Applications," (volume 6, number 8) has a comprehensive description of graphics standards.

CGM

CGM is ANSI standard ANSI X3.122-1986 and can be obtained from the ANSI address listed above. The CGM is a description of functional elements and encodings. Some information on the standard is contained in "The NCAR Computer Graphics Metafile Translator" section of the *NCAR Graphics Installer’s Guide*. 

August 1987
What to Do if You Have Questions or Suggestions

If you have questions about using NCAR Graphics, contact your NCAR Graphics site representative. If you have suggestions about this manual, please fill out the form that is included as the last page in this manual, or send a letter to:

NCAR/Scientific Computing Division
Documentation Group
User Services
P.O. Box 3000
Boulder, CO 80307-3000
Section II: Guide for Converting Code from the NCAR System Plot Package to the NCAR GKS Graphics Package

Introduction ................................................................. 21

System Plot Package Simulator (SPPS) .................................. 21
  Log Scaling ............................................................. 21

The Six Conversion Steps .................................................. 22
  Step 1: Convert to FORTRAN 77 ........................................... 22
  Step 2: Eliminate Absolute Integer Addressing .......................... 22
  Step 3: Make Necessary Changes to NSPP Calls ....................... 24
    Use FINDG to Find NSPP Calls in Your Code ....................... 26
  Step 4: Make Changes to Calls to Higher-level Utilities .......... 26
    Changes in Coordinate Addressing .................................. 27
    Changes in Character Manipulation ................................ 27
    Changes to PWRTX Options ........................................... 27
    Changes to CONRAN, CONRAQ, and CONRAS Options ............ 27
    Changes to DASHD Options .......................................... 28
    Changes to PWRM Options ............................................ 28
  Step 5: Eliminate Obsolete Support Routines .......................... 28
  Step 6: Insert Necessary Open and Close Calls ...................... 29
    Open GKS and Open and Activate a Workstation ................. 29
    Deactivate and Close the Workstation and Close GKS .......... 30

Conversion Example ....................................................... 30
  The NSPP-based Code ................................................ 30
  Conversion to NCAR Graphics Code .................................... 32
    Step 1 .................................................................. 32
    Step 2 .................................................................. 32
    Step 3 .................................................................. 32
    Step 4 .................................................................. 32
    Step 5 .................................................................. 33
    Step 6 .................................................................. 33
  The Completed NCAR Graphics Code .................................... 33

Former Routines and Utilities Not in NCAR Graphics ............... 34
  The Former WINDOW Utility and Clipping in NCAR Graphics .... 34
  The Former SCROLL and the FLASH Routines ....................... 35
  The Former CONRECSMTH Utility .................................... 35
  The Former SUPMAP Utility ............................................ 35

Appendix A: Notes on FORTRAN 77 TYPE CHARACTER Variables ... 37
Introduction

This section is a guide for converting program units that use the FORTRAN 66 NCAR System Plot Package and the FORTRAN 66 higher-level utilities to program units that use GKS and the FORTRAN 77 higher-level utilities. This manual section contains

- the step-by-step sequence of how to convert NSPP-based code to FORTRAN 77 GKS-based code.
- an example of one complete conversion.
- a description of the GKS clipping function and how to turn it off.
- an appendix of notes about FORTRAN 77 CHARACTER variables.

The GKS-based NCAR Graphics package has essentially the same functionality as the NSPP-based graphics package. Almost all of the NCAR higher-level FORTRAN graphics utilities have been converted to a FORTRAN 77 package compatible with GKS Level 0A. However, WINDOW, SCROLL, and FLASH are not part of NCAR Graphics; if you were using these under NSPP, you should read about them at the end of this section. Also, read the end of this section concerning the former CONRECSMTH and SUPMAP utilities.

If you have not used any previous NCAR graphics packages, then you do not need to read this section of the manual.

System Plot Package Simulator (SPPS)

Because eliminating all NSPP calls in favor of GKS calls would be a difficult task — especially since the functionality of GKS 0A and NSPP is not identical — a package of GKS-based subroutines that emulates NSPP has been written. This package is called the System Plot Package Simulator, SPPS. Not all entries of NSPP appear in SPPS, and the use of some of the entries has changed even though the entries have the same name. You should familiarize yourself with SPPS by reading the "SPPS" section of this manual. Be sure to read the "SPPS Subroutines and Their Relationship to GKS" part in the "SPPS" section of this manual.

Log scaling

GKS does not provide facilities for automatic log scaling. However, since approximately 20% of all graphics jobs that run at NCAR use a non-default value for the log-scaling parameter of the SET call, an automatic log-scaling
scheme has been implemented for use with the higher-level NCAR Graphics utilities.

The only way to use this automatic log-scaling scheme is via the SPPS entry SET. The final argument to the SPPS entry SET still produces the same results in the GKS-based utilities that it did in the NSPP-based utilities.

It is important to note that the final argument in the SET call has no effect on direct GKS calls, such as GPL, GPM, and GTX. The final argument to SET governs only what is produced by the NCAR Graphics utilities. If you desire log scaling when using any direct GKS call, you will have to do the scaling yourself before making the direct GKS call.

The Six Conversion Steps

To convert a given NSPP-based program unit to NCAR Graphics, you must perform the following tasks:

1. Convert to FORTRAN 77.
2. Eliminate the absolute integer addressing allowed in NSPP but not allowed in GKS.
3. Make necessary changes to any NSPP calls.
4. Make appropriate changes in calls to higher-level utilities.
5. Eliminate obsolete support routines that were called directly.
6. Insert calls to open GKS, open and activate a workstation, close and deactivate a workstation, and close GKS.

A detailed discussion of each step follows.

Step 1: Convert to FORTRAN 77

Converting to FORTRAN 77 is a matter of performing the language conversion from FORTRAN 66. Probably the most significant change from FORTRAN 66 to FORTRAN 77 is the introduction of the TYPE CHARACTER variable. This represents an incompatibility between FORTRAN 66 and FORTRAN 77; the Hollerith "nH" designator allowed in FORTRAN 66 is not allowed in FORTRAN 77 (although many compilers accept it as an extension to FORTRAN 77).

If you need help in converting to usage of TYPE CHARACTER variables, see Appendix A at the end of this section, or consult the FORTRAN standard. The FORTRAN standard is ANSI X3.9-1978. To order, write to: ANSI, 1430 Broadway, New York, NY 10018, or call (212) 354-3300.

Step 2: Eliminate Absolute Integer Addressing

The NSPP-based package allowed for two different forms of coordinate addressing — absolute and world coordinate. Absolute coordinates are integer coordinates. In the NSPP-based package, the absolute integer coordinates...
referenced positions on an abstract plotting screen and by default were in the range 1 to 1024.

*World coordinates* are also known as user coordinates or user space. World coordinates are defined in NSPP by the second set of four arguments in a SET call. These are the same coordinates that are required in a call to the GKS function, SET WINDOW (GSWN).

Absolute integer addressing is not allowed in the GKS-based NCAR Graphics package. This restriction has been imposed by the conversion to FORTRAN 77, which does not allow the same argument to have two different types. Absolute integer addresses were used in the NSPP-based package in one of two ways:

1. As absolute integer addresses on a virtual screen.
2. As absolute lengths for lines and sizes for characters.

Plotter Address Units (PAUs) used for absolute addressing, as in item 1 above, have been eliminated from NCAR Graphics. PAUs used as lengths, as in the second item, have been retained. This was done for the following reasons. Since GKS has no facility for absolute addressing in the sense of item 1, any such addressing in the GKS utilities must be implemented in the software by using some mechanism such as the INTT function (a function to determine if an argument is an integer or a floating-point value). INTT is impossible to implement on some computers. Also, FORTRAN 77 requires that all actual arguments agree in type with the corresponding dummy argument in a subroutine or function call, and this disallows using the same argument for both floating-point and integer values. To allow for integer addressing on a virtual screen, in addition to addressing in world coordinates, it would be necessary to have distinct entry points for integer and floating parameters, and this would introduce more confusion than the reward would warrant.

PAUs are used extensively throughout NSPP as absolute lengths, as in item 2. For example, they are used as arguments to PWRIT, PWRITX, GRIDAL, and DASHD, among others. They are used internally in many places as well. So, retaining the concept of PAUs as absolute lengths is appropriate; otherwise, the argument lists to many routines would have to be changed.

Please note the following when selecting the proper lengths as defined in item 2. The length of a PAU in NSPP was determined by the current scale factors set by SETI, and the default size was 1/1024th of the maximum screen size. If the virtual screen resolution had been changed to 512 x 512 with a call to SETI, then one PAU would have a length of 1/512th of the maximum screen size.

In NCAR Graphics, two new parameters, XF and YF, have been added to the list of user-definable utility state variables in the subroutines SETUSV and GETUSV (see the "SPPS" section for details). The purpose of the XF and YF parameters is identical to that of the arguments in the old SETI subroutine. When determining the width of a line or the height of a
character specified in PAUs, you will need to check the values of XF and YF to correctly compute the fraction of the screen size.

Step 3: Make Necessary Changes to NSPP Calls

There are some subtle differences in the results produced by some SPPS entries and the results produced by the corresponding NSPP entry. By far the largest difference is that absolute integer coordinate addressing is not allowed in SPPS. A complete set of coordinate conversion routines is described in the "SPPS" section of this manual, which you should read if you are using absolute integer coordinate addressing. To get the world-coordinate address, all you need to do is apply the appropriate conversion function from SPPS.

Table 2 lists the NSPP entry points and their status in SPPS.
### Table 2 — NSPP and SPPS Entry Points

<table>
<thead>
<tr>
<th>NSPP</th>
<th>Status in SPPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURVE</td>
<td>*Valid call, check SPPS for differences</td>
</tr>
<tr>
<td>FL2INT</td>
<td>No differences at all</td>
</tr>
<tr>
<td>FLASH1</td>
<td>No FLASH entries allowed, emulation of these</td>
</tr>
<tr>
<td>FLASH2</td>
<td>subroutines would require a GKS package at</td>
</tr>
<tr>
<td>FLASH3</td>
<td>Level 2A or higher</td>
</tr>
<tr>
<td>FLASH4</td>
<td></td>
</tr>
<tr>
<td>FLUSH</td>
<td>No differences at all</td>
</tr>
<tr>
<td>FRAME</td>
<td>*Valid calls,</td>
</tr>
<tr>
<td>FRSTPT</td>
<td>check SPPS for differences</td>
</tr>
<tr>
<td>GETCND</td>
<td></td>
</tr>
<tr>
<td>GETSI</td>
<td>No differences at all</td>
</tr>
<tr>
<td>GETOPT</td>
<td>Not in SPPS, some functions replaced by SETUSV</td>
</tr>
<tr>
<td>GRID</td>
<td>All collected into GRIDAL,</td>
</tr>
<tr>
<td>GRIDAL</td>
<td>see GRIDAL in the &quot;Utilities&quot; section</td>
</tr>
<tr>
<td>GRIDL</td>
<td></td>
</tr>
<tr>
<td>HALFAAX</td>
<td></td>
</tr>
<tr>
<td>LABMOD</td>
<td></td>
</tr>
<tr>
<td>LINE</td>
<td>*Valid calls,</td>
</tr>
<tr>
<td>MXMY</td>
<td>check SPPS for differences</td>
</tr>
<tr>
<td>OPTN</td>
<td>Not in SPPS, some functions replaced by SETUSV</td>
</tr>
<tr>
<td>PERIM</td>
<td>Collected into GRIDAL, see GRIDAL</td>
</tr>
<tr>
<td>PERIML</td>
<td>in &quot;Utilities&quot; section</td>
</tr>
<tr>
<td>PLOTIT</td>
<td>*Valid calls,</td>
</tr>
<tr>
<td>POINT</td>
<td>check SPPS for differences</td>
</tr>
<tr>
<td>POINTS</td>
<td></td>
</tr>
<tr>
<td>PWRT</td>
<td></td>
</tr>
<tr>
<td>SET</td>
<td></td>
</tr>
<tr>
<td>SETCND</td>
<td>Not in SPPS, some functions replaced by SETUSV</td>
</tr>
<tr>
<td>SETI</td>
<td>*Valid call, check SPPS for differences</td>
</tr>
<tr>
<td>TICK4</td>
<td>Collected into GRIDAL, see GRIDAL in &quot;Utilities&quot;</td>
</tr>
<tr>
<td>VECTOR</td>
<td>*Valid call, check SPPS for differences</td>
</tr>
</tbody>
</table>

*Requires world coordinates
Use FINDG to Find NSPP Calls in Your Code

Before you can make changes to NSPP calls, you have to be able to find them in your code. Your NCAR Graphics site representative should have supplied the FINDG utility for this purpose. FINDG reads the FORTRAN utility you wish to convert as input from logical unit 5 and writes its output to logical unit 6. Whenever FINDG locates an NSPP entry in a line of the input FORTRAN file, it copies the line number and code onto the output file. The only exception is that if any line beginning with a "$" symbol is encountered, then a list of the names being searched for is printed and execution stops. FINDG also locates and echoes lines that contain calls to certain higher-level utilities such as the CONOP routines of CONRAN.

FINDG searches for the following names:

<table>
<thead>
<tr>
<th>AXES</th>
<th>FLUSHB</th>
<th>OPTN</th>
<th>PWRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONOP1</td>
<td>FRAME</td>
<td>PACKUM</td>
<td>PWRT</td>
</tr>
<tr>
<td>CONOP2</td>
<td>FRST3</td>
<td>PERIM3</td>
<td>PWRX</td>
</tr>
<tr>
<td>CONOP3</td>
<td>FRSTD</td>
<td>PERROR</td>
<td>PWRY</td>
</tr>
<tr>
<td>CONOP4</td>
<td>FRSTPT</td>
<td>PLOTIT</td>
<td>SET</td>
</tr>
<tr>
<td>CURVE</td>
<td>GETCHR</td>
<td>POINT</td>
<td>SET3</td>
</tr>
<tr>
<td>CURVE3</td>
<td>GETCND</td>
<td>POINT3</td>
<td>SETCHR</td>
</tr>
<tr>
<td>CURVED</td>
<td>GETOPT</td>
<td>POINTS</td>
<td>SETCND</td>
</tr>
<tr>
<td>DASHLN</td>
<td>GETSET</td>
<td>PORGN</td>
<td>SETI</td>
</tr>
<tr>
<td>DASHD</td>
<td>GETSI</td>
<td>PREOUT</td>
<td>TICK3</td>
</tr>
<tr>
<td>ENCODE</td>
<td>INTT</td>
<td>PSCALE</td>
<td>TICK43</td>
</tr>
<tr>
<td>FENCE3</td>
<td>JUSTFY</td>
<td>PSYM</td>
<td>TRANS</td>
</tr>
<tr>
<td>FL2INT</td>
<td>LINE</td>
<td>PSYM3</td>
<td>ULIBER</td>
</tr>
<tr>
<td>FLASH1</td>
<td>LINE3</td>
<td>PUT42</td>
<td>VECT3</td>
</tr>
<tr>
<td>FLASH2</td>
<td>LINED</td>
<td>PUTINS</td>
<td>VECTD</td>
</tr>
<tr>
<td>FLASH3</td>
<td>LOC</td>
<td>PWRT</td>
<td>VECTOR</td>
</tr>
<tr>
<td>FLASH4</td>
<td>MXMY</td>
<td>PWRTX</td>
<td>WRITEB</td>
</tr>
<tr>
<td>FLUSH</td>
<td>OPTION</td>
<td>PWRTY</td>
<td></td>
</tr>
</tbody>
</table>

To display this list on-line, place a "$" as the first character of any line, and the list will be displayed.

Note that the utility searches only for calls; comment statements that contain the searched-for name will not appear in the output list.

Once you have located the appropriate calls, proceed as advised in the "SPPS" section if the subroutine called is a part of the SPPS package. If the call is not part of SPPS, continue to follow the steps listed here.

Step 4: Make Changes to Calls to Higher-level Utilities

A primary goal in converting the higher-level utilities to FORTRAN 77 was to retain as much similarity as possible between the argument lists in NCAR Graphics and the argument lists in the NSPP-based package. However, the calls to higher-level utilities had to be changed in two significant ways:
1. The coordinate addressing in NCAR Graphics is strictly in world coordinates.

2. All character manipulations in NCAR Graphics are handled by FORTRAN 77 TYPE CHARACTER variables.

**Changes in Coordinate Addressing:** The alternate absolute coordinate addressing allowed in NSPP is not available in NCAR Graphics. To ease the transition from one coordinate space to another, all possible coordinate space conversion functions are contained in the SPPS package. Therefore, if you were previously using absolute coordinate specifications in a call to a utility, then those coordinate specifications must be replaced with the appropriate function applied to the old coordinates. Read the "SPPS" section to learn how to do this. Also, see the conversion example at the end of this section.

**Changes in Character Manipulation:** The second significant change is that FORTRAN 77 TYPE CHARACTER variables are used to perform character manipulations in NCAR Graphics. All character information passed to the utilities must use quoted strings, or CHARACTER variables, not integer or floating-point variables.

**Changes to PWRITX Options:** Examine, for example, the effect of coordinate and character information changes in the subroutine PWRITX. The first two arguments to PWRITX specify the coordinate position of the character string to follow, and they must be specified as world coordinates; the third argument, which specifies the characters to be drawn, must be a quoted string or a TYPE CHARACTER variable. The character count parameter in PWRITX has been retained to avoid changing the argument list, even though it could have been removed since it is possible to determine the length of a character string by using the LEN function in FORTRAN 77.

Here is an example call to PWRITX, using the pre-GKS FORTRAN 66 package:

```fortran
CALL PWRITX(500,450,29,'THIS IS 'U' S' S'UPERSCRIPTING',29,0,0,1)
```

Here is the equivalent call using the GKS-based FORTRAN 77 package:

```fortran
CALL PWRITX(CPUX(500),CPUY(450),
             'THIS IS '"U"'S"S"UPERSCRIPTING',29,0,0,1)
```

**Changes to CONRAN, CONRAQ, and CONRAS Options:** The option-setting routines of the CONRAN, CONRAQ, and CONRAS packages have been changed to make them conform to FORTRAN 77. All calls to CONOP1, CONOP2, CONOP3, or CONOP4 should be examined against the NCAR Graphics documentation for CONRAN, CONRAQ, and CONRAS. To locate CONOP calls, use the FINDG utility, which is described earlier in this section.
Changes to DASHD Options: NSPP had DASHD arguments that allowed specification of dash patterns as either character or integer values. To preserve this functionality in FORTRAN 77, DASHD has been split into two separate subroutines:

SUBROUTINE DASHDB (IPAT)

in which IPAT is an integer representation for the dash pattern defined in exactly the way it was in the old DASH utilities (DASHLINE, DASHCHAR, DASHSMTH, DASHSUPR), and

SUBROUTINE DASHDC (IPAT, JCRT, JSIZE)

in which IPAT is now a FORTRAN 77 TYPE CHARACTER string, and JCRT, and JSIZE are defined as they were in the old DASH utilities. Note, however, that JCRT and JSIZE must always be integers.

Changes to PWRM Options: Another argument list that has been changed for FORTRAN 77 compatibility is that of PWRM (an entry point in DASHSUPR). The new entry is PWRTM:

SUBROUTINE PWRTM (X,Y,IDPC,ISIZ,IOR,ICNT)

The arguments for PWRTM are identical to those for the SPPS routine WTSTR. For details on WTSTR, see the "SPPS" section.

Step 5: Eliminate Obsolete Support Routines

This task is necessary only if you directly call any of the 14 locally-implemented support routines for NSPP. Nine of the 14 are not in NCAR Graphics because of portability or other problems. The deleted routines are as follows:

ENCODE       PERRROR
GETCHR        SETCHR
INTT          ULIBER
LOC           WRITEB
PACKUM

As these routines were intended as system-level calls, most users should never have been calling them. However, if you were calling any of these routines directly and need help in determining what calls to use as replacements, contact your NCAR Graphics site representative.
NCAR Graphics utilizes the following support routines:

- GBYTES
- G01MIO
- IAND
- IOR
- ISHIFT
- I1MACH
- R1MACH
- SBYTES

Most users do not need to call these support routines directly. However, if you need more details on these, consult the NCAR Graphics Installer's Guide.

Step 6: Insert Necessary Open and Close Calls

Unlike previous NCAR graphics packages, the GKS-based NCAR Graphics package has to be opened before use and closed after use. The GKS standard abstracts the common definition of a workstation to include a metafile as a special type of workstation. This type of workstation is called a Metafile Output (MO) workstation. If you are using NCAR's GKS Level 0A, the only allowable workstation is an MO workstation.

If you are using a higher level of GKS implementation, several workstations can be open and active simultaneously. For example, you can capture a plot in a metafile at the same time you are seeing it drawn on your screen.

In addition to opening the GKS package itself, individual workstations must be opened and activated.

Open GKS and Open and Activate a Workstation: All GKS calls begin with the letter G. To open GKS, issue the GKS call:

```fortran
CALL GOPKS (IER, ISZ)
```

IER (usually the standard output unit 6) is the logical unit to receive GKS error messages. ISZ is the maximum buffer size allowed by GKS. In the NCAR GKS package, ISZ is a dummy argument, but it must appear in the argument list. If you are using another GKS package, consult your NCAR Graphics site representative to determine what ISZ should be.

To open a workstation, issue the call:

```fortran
CALL GOPWK (1, 2, 1)
```

This call establishes the workstation identifier as 1, the FORTRAN logical unit number for output as 2, and the workstation type as 1 (the value for the type of the Metafile Output workstation in the NCAR GKS 0A package). With the routine G01MIO that is provided automatically in the NCAR GKS package, the invocation GOPWK will open FORTRAN logical unit 2 with file name GMETA.
To activate the workstation, issue the call:

CALL GACWK (1)

If you are using NCAR GKS 0A, you can use

CALL OPNGKS

in place of the three calls described above. OPNGKS is part of SPPS.

**Deactivate and Close the Workstation and Close GKS:** When you are finished using NCAR Graphics, the workstation must be deactivated and closed and GKS must be closed. To deactivate and close the workstation, issue the calls:

CALL GDAWK (1)

CALL GCLWK (1)

where 1 is the workstation identifier. To close GKS, issue the call:

CALL GCLKS

If you are using NCAR GKS 0A, you can use

CALL CLSGKS

in place of the three calls described above. CLSGKS is part of SPPS.

When you have finished these conversion steps, your former NSPP-based code is ready to run under NCAR Graphics. A complete example of one case follows.

**Conversion Example**

The following conversion from an NSPP-based code to a code using NCAR Graphics presents a particularly complex example to illustrate most of the conversion steps.

**The NSPP-based Code**

```c
C Set up CONRAN data.

DIMENSION WK(221), IWK(744), SCR(1600)
DIMENSION XD(17), YD (17), ZD (17), XP (4), YP (4)
DATA XD(1), XD(2), XD(3), XD(4), XD(5), XD(6), XD(7), XD(8),
1 XD(9), XD(10), XD(11), XD(12), XD(13), XD(14), XD(15),
2 XD(16), XD(17)
3 /3., 3., 10., 18., 18., 10., 10., 5., 1., 15., 20.,
4 5., 15., 10., 7., 13., 16. /
DATA YD (1), YD(2), YD(3), YD(4), YD(5), YD(6), YD(7), YD(8),
```
Define coordinates for drawing points.

```
1 YD(9), YD(10), YD(11), YD(12), YD(13), YD(14), YD(15),
2 YD(16), YD(17)
3 /3., 18., 18., 3., 18., 10., 1., 5., 10., 5., 10.,
4 15., 15., 15., 20., 20., 8. /
DATA ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6), ZD(7), ZD(8),
1 ZD(9), ZD(10), ZD(11), ZD(12), ZD(13), ZD(14), ZD(15),
2 ZD(16), ZD(17)
4 1., 1., 1., 1., 1., 25. /
DATA NDP/17/
```

Define window, viewport, and log scaling.

```
CALL SET(0., 1., 0., 1., 0., 1., 0., 1., 1)
```

Draw boundary line along left edge.

```
CALL LINE (1,1,1,1024)
```

Draw four asterisks at top of plot.

```
CALL POINTS(XP,YP,4, 1H*,0)
```

Establish dash pattern and draw dashed line near right edge.

```
CALL DASHD(9H$$''X'',9,8,10)
CALL LINED(1000,1,1000,1024)
```

Establish CONRAN options, and call CONRAN to draw contours.

```
CALL CONOP3(6HCHL=ON, 20., 0.)
CALL CONOP1(6HEXT=ON)
CALL CONOP3(6HTLE=ON, 11HCONRAN DEMO, 11)
CALL CONOP1(7HTRI=OFF)
CALL CONRAN(XD,YD,ZD,NDP,WK,IK,WK,SCR)
CALL FRAME
```

Print terminating message.

```
WRITE (6,10)
10 FORMAT(1X,13HPLOT COMPLETE)
```

STOP
END
Conversion to NCAR Graphics Code

**Step 1:** Convert the code to FORTRAN 77. This requires that the Hollerith constants in the calls to POINTS, DASHD, CONOP1, and CONOP3 and the Hollerith constants in the FORMAT statement be changed to character strings.

**Step 2:** Eliminate any absolute coordinate addresses in favor of world coordinates. Passing the input file through FINDG results in the following output:

```plaintext
LINE 34: CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
LINE 38: CALL LINE (1,1,1,1024)
LINE 42: CALL POINTS(XP,YP,4,'H*','0)
LINE 46: CALL DASHD('9H$$$''X'',9,8,10)
LINE 47: CALL LINED(1000,1,1000,1024)
LINE 51: CALL CONOP3('6HCHL=ON,20.,0.2')
LINE 52: CALL CONOP1('6HEXT=ON')
LINE 53: CALL CONOP3('6HTLE=ON,11HCONRAN DEMO,11')
LINE 54: CALL CONOP1('7HTRI=OFF')
LINE 56: CALL FRAME

END OF FILE
TOTAL NUMBER LINES READ: 64
```

This represents a list of all subroutine calls that may need to be changed. Note that the coordinates in the calls to LINE and LINED need to be changed. Convert these coordinates in one of two ways: either take the specified coordinate and convert it to the required world coordinate, or use the conversion functions supplied as part of the SPPS package. For example, the argument 1000 could be changed directly to .9765 (=(1000-1)/1023) or the SPPS conversion function CPUX could be applied. In the long run, it will probably be easier to use the SPPS conversion routines to avoid having to perform the division each time yourself.

**Step 3:** Make changes to NSPP calls, which is easy in this case since there are none. Note that the call to SET can remain unchanged to get the appropriate log scaling.

**Step 4:** Make changes to calls to higher-level utilities. Modify the CONOP calls of the CONRAN utility. (They were changed to conform to FORTRAN 77.) Also, the call to POINTS is slightly different in NCAR Graphics, so that needs to be corrected.
Step 5: Eliminate outmoded support routines. Nothing has to be done since there are no direct calls to any support routines.

Step 6: Insert the required open and close GKS and workstation calls.

The Completed NCAR Graphics Code

After completing these steps, the converted code emerges:

C
C Set up CONRAN data.
C
DIMENSION WK(221), IWK(744), SCR(1600)
DIMENSION XD(17), YD(17), ZD(17), XP(4), YP(4), AR(2)
DATA XD(1), XD(2), XD(3), XD(4), XD(5), XD(6), XD(7), XD(8),
1 XD(9), XD(10), XD(11), XD(12), XD(13), XD(14), XD(15),
2 XD(16), XD(17)
3 /3.,3.,10.,18.,18.,10.,10.,5.,1.,15.,20.,
4 5.,15.,10.,7.,13.,16./
DATA YD(1), YD(2), YD(3), YD(4), YD(5), YD(6), YD(7), YD(8),
1 YD(9), YD(10), YD(11), YD(12), YD(13), YD(14), YD(15),
2 YD(16), YD(17)
3 /3.,18.,18.,3.,18.,10.,1.,5.,10.,5.,10.,
4 15.,15.,15.,20.,20.,8./
DATA ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6), ZD(7), ZD(8),
1 ZD(9), ZD(10), ZD(11), ZD(12), ZD(13), ZD(14), ZD(15),
2 ZD(16), ZD(17)
4 1.,1.,1.,1.,1.,25./
DATA NDP/17/

C
C Define coordinates for drawing points.
C
DATA XP(1), XP(2), XP(3), XP(4),.2,.25,.75,.8/
DATA YP(1), YP(2), YP(3), YP(4),.975,.975,.975,.975/

C
C Open GKS, open workstation of type 1, activate workstation.
C
CALL OPNGKS

C
C Define window, viewport, and log scaling.
C
CALL SET(0.,1.,0.,1.,0.,1.,1.,1.)

C
C Draw boundary line along left edge.
C
CALL LINE (CPUX(1), CPUY(1), CPUX(1), CPUY(1024))

C
C Draw four asterisks at top of plot.
C
CALL POINTS(XP, YP, 4, 42, 0)

C
C Establish dash pattern and draw dashed line near right edge.
CALL DASHDC('$$$$''''X''''',8,10)
CALL LINED(CPUX(1000),CPUY(1),CPUX(1000),CPUY(1024))

C Establish CONRAN options, and call CONRAN to draw contours.
C
AR(1) = 20.
AR(2) = 0.
CALL CONOP3('CHL=ON',AR,2)
CALL CONOP1('EXT=ON')
CALL CONOP4('TLE=ON','CONRAN DEMO',11,0)
CALL CONOP1('TRI=OFF')
CALL CONRAN(XD,YD,ZD,NDP,WK,IWK,SCR)
CALL FRAME

C Deactivate and close workstation, close GKS.
C
CALL CLSGKS

C Print terminating message.
C
WRITE(6,10)
10 FORMAT(' PLOT COMPLETE')
C
STOP
END

Former Routines and Utilities Not in NCAR Graphics

The Former WINDOW Utility and Clipping in NCAR Graphics

The old WINDOW utility is not a part of NCAR Graphics. The function performed by WINDOW is known as clipping — cutting away all parts of a picture that lie outside of the user-defined window. Under the NSPP package, the user-defined window (sometimes also called user space or world coordinate space) is the rectangular space defined by the second set of four arguments in a SET call. Since clipping is a part of GKS functionality, the WINDOW package is unnecessary.

One significant difference between NCAR Graphics and NSPP is that by default GKS clips to the user-defined window. For example, if the following call were made

CALL SET(.1,.9,.1,.9,10.,100.,5.,.50.,1)

any portion of the plot that was outside of the rectangle with corner points (10,5) and (100,50) would not be displayed. You can turn clipping off by making the GKS call:

CALL GSCLIP(0)
The Former SCROLL and the FLASH Routines

The higher-level utility for producing scrolled movie titles, SCROLL, does not appear in NCAR Graphics. SCROLL depends on the NSPP FLASH routines. The FLASH logic capability is not included in the NCAR Graphics implementation because there is no easy way to emulate the FLASH logic at Level 0A in GKS. However, if you are using a GKS package that implements the segmentation functions (GKS level 2B or higher), then emulation of the FLASH package is a straightforward matter. Use the GKS documentation for level 2B or above to learn how to emulate FLASH.

The Former CONRECSMTH Utility

The former CONRECSMTH utility is not included in NCAR Graphics. Since CONRESCMTH=CONREC+DASHSMTH, it is not necessary to maintain CONRECSMTH as a separate file.

The Former SUPMAP Utility

The SUPMAP package in previous non-GKS versions of NCAR graphics has been replaced by EZMAP. EZMAP has a different user interface than SUPMAP and contains more features. EZMAP does retain a SUPMAP entry in support of those users who still want to call SUPMAP.
Appendix A: Notes on FORTRAN 77 TYPE CHARACTER Variables

You will probably encounter problems with handling characters when converting FORTRAN 66 files to FORTRAN 77. In the FORTRAN 66 standard, characters (using the Hollerith specifier “nH”) and character data were stored in integer variables and arrays. For example, executing the FORTRAN 66 statements

\[ \begin{align*}
J &= 2 \\
\text{WRITE (6, 500)} \\
500 & \quad \text{FORMAT (3H J=, I1)}
\end{align*} \]

produces

\[ J=2 \]

on the standard output device. To use another example of FORTRAN 66, let \text{SUB(I)} be a FORTRAN 66 subroutine having one integer argument. The statement

\[ \text{CALL SUB(2HAB)} \]

is a legal FORTRAN 66 call in which the variable I in the subroutine \text{SUB} assumes the internal character representation of the string "AB".

Because different computers can have different word lengths and different internal representations for characters (both in terms of encoding and the character length), storing character data in integer variables and arrays is unsatisfactory. In FORTRAN 66, character data were and frequently still are handled in an \textit{ad hoc} and non-portable manner. Thus, the \textbf{TYPE CHARACTER} statement has been added to the FORTRAN standard.

The \textbf{TYPE CHARACTER} statement represents one of the greatest differences between the two standards, since its inclusion in FORTRAN 77 is not upwardly-compatible with FORTRAN 66. The "nH" specifier is not allowed in FORTRAN 77, although many compilers accept it as a non-standard extension. For example, CFT, the CRAY FORTRAN compiler, accepts the "nH" specifier on CRAY computers. If you need complete details about "nH", see Appendix C of the FORTRAN 77 standard. (The FORTRAN standard is ANSI X3.9-1978. To order, write: ANSI, 1430 Broadway, New York, NY 10018, or call 212-354-3300.)

Before illustrating how to convert FORTRAN 66 Hollerith data items to FORTRAN 77 constructs, it is important to examine the FORTRAN 77 \textbf{TYPE CHARACTER} statement. If you need the full definition of the FORTRAN 77 \textbf{TYPE CHARACTER} statement, see the FORTRAN 77 Standard, Section 8.4.2, pgs. 8-5.

To illustrate the use of the FORTRAN 77 \textbf{TYPE CHARACTER} statement, consider the following examples.
Example 1: In the simplest instance, consider the statement

```
CHARACTER*3 A
```

that flags "A" as a TYPE CHARACTER variable. The compiler reserves enough space on the given computer to store 3 characters in the variable "A." To assign character information to "A," form a statement like the one that follows:

```
A = 'ABC'
```

Thus, executing the FORTRAN 77 statements

```
CHARACTER*3 A
A = 'ABC'
WRITE(6,500) A
500 FORMAT(' A='A3)
```

produces

```
A=ABC
```

on any computer running a FORTRAN 77 compiler. If you examine the FORTRAN 66 equivalent to the above example, you can see the superiority of FORTRAN 77 to FORTRAN 66 in the handling of characters. The statement A=3HABC, for example, may not be legal on some computers using FORTRAN 66 without dimensioning the variable A (and typing "A" INTEGER as well), since the computer may only allow 2 characters per word. The standards are intended to eliminate the necessity of performing different operations when handling the same situation on different computers.

Example 2: Now consider another example of TYPE CHARACTER statements. The statement

```
CHARACTER*5 I (2)
```

indicates that the array "I" is of TYPE CHARACTER, and each array element is 5 characters long. Thus the compiler reserves space for a total of 10 characters for the "I" array. Executing the statements

```
CHARACTER*5 I (2)
I (1) = 'ABCDE'
I (2) = '12345'
WRITE (6,500) I (1),I (2)
500 FORMAT (' I='2A5)
```

produces

```
I=ABCDE12345
```

August 1987
An equivalent way of doing this would be to execute the following statements:

```fortran
CHARACTER*10 I
I = 'ABCDE12345'
WRITE(6,500) I
500 FORMAT(' I='A10)
```

**Example 3:**
Consider yet another example.

```fortran
CHARACTER*3 A,B*2,C
```

with type "A" as a character variable of length 3, "B" as a character variable of length 2, and "C" as a character variable of length 3.

You can select specific characters, sections of characters from character variables, or arrays by using the structure \( (n:m) \) to indicate the characters you desire. For example, the code

```fortran
CHARACTER*4 J,A(2)
CHARACTER*2 B,C
DATA J,A(1),A(2)/'LMNO', 'ABCD', '5678'/
B = J(2:3)
C = A(2) (2:3)
WRITE(6,500) B,C
500 FORMAT(' B=',A2/' C='A2)
```

produces

```
B=MN
C=67
```

on the standard output. The character variable "B" has been set to the second and third characters of the character variable "J"; the character variable "C" has been set to the second and third characters of the second array element of the character array "A".

**Example 4:**

Some intrinsic functions useful in dealing with character variables are LEN, ICHAR, and CHAR, which are part of the FORTRAN 77 standard. The function LEN returns the length of a given character variable; the function ICHAR converts a character variable to its internal representation integer equivalent; the function CHAR converts an internal integer representation to a character and sets a character variable to the character so represented. For example, in the ASCII character set, the character "A" is represented internally by the integer 65. Executing the code

```fortran
CHARACTER*5 X
CHARACTER*1 Y
J = LEN(X)
K = ICHAR ('A')
Y = CHAR(K)
WRITE(6,500) J,K,Y
500 FORMAT(' J=',I1, ' K=',I2, ' Y='A1)
```

August 1987
produces

\[ J = 5, \; K = 65, \; Y = A \]

on the standard output.

**Example 5:**
Also, concerning character manipulation, you may need to know how to handle characters as arguments to subroutines when you don't know before calling the subroutine how long the input string will be. Use the construct `CHARACTER*(*)` in the subroutine to assist you in this situation. Note the use of the `CHARACTER*(*)` construct in the following example:

Executing the program

```fortran
CHARACTER*3 A
CHARACTER*5 B
DATA A, B/'ABC', 'DEFGH'/
CALL PS(A)
CALL PS(B)
STOP
END
SUBROUTINE PS(TT)
  CHARACTER*(*) TT
  L = LEN(TT)
  WRITE(6,500) L
  FORMAT(' LENGTH = 'I3)
END
```

produces

```
LENGTH = 3
LENGTH = 5
```

on the standard output.

**Example 6:**
If a character variable is set to a string longer than its length, then it is truncated. If a character variable is set to a string shorter than its length, then it is padded with blanks. For example, executing the code

```fortran
CHARACTER*3 A
A = 'ABCD'
WRITE(6,500) A
A = 'ST'
WRITE(6,500) A
500 FORMAT(' A = 'A3)
```

produces

```
A = ABC
A = ST
```

on the standard output.
You should also be aware of how to use the concatenation operator "//" on character variables. For instance, the code

```fortran
CHARACTER*4 A
CHARACTER*2 B
CHARACTER*6 C
A = 'ABCD'
B = 'ST'
C = A // B
WRITE(6,500) C
500 FORMAT(' C = 'A6)
```

produces

```
C = ABCDST
```

Section III: SPPS — An NCAR System Plot Package Simulator

Introduction ................................................................................................................. 45
Coordinate Systems ..................................................................................................... 46
Coordinate-system Mapping Functions ........................................................................ 47

SPPS Subroutines and Their Relationship to GKS ...................................................... 48
 CLSGKS ...................................................................................................................... 48
 CURVE (PX,PY,NP) ................................................................................................. 48
 FLUSH ......................................................................................................................... 48
 FL2INT (PX,PY,IX,IY) ............................................................................................. 49
 FRAME ......................................................................................................................... 49
 FRSTPT (PX,PY) ......................................................................................................... 49
 GETSET (FL,FR,FB,FT,UL,UR,UB,UT,LL) ............................................................. 49
 GETSI (IX,IY) ............................................................................................................ 50
 GETUSV (VN,IV) ....................................................................................................... 50
 LINE (X1,Y1,X2,Y2) ............................................................................................... 50
 MXMY (IX,IY) ........................................................................................................... 50
 OPNGKS ..................................................................................................................... 50
 PLOTIF (FX,FY,IP) ................................................................................................... 51
 PLOTIT (IX,IY,IP) ..................................................................................................... 51
 POINT (PX,PY) .......................................................................................................... 51
 POINTS (PX,PY,NP,IC,IL) ....................................................................................... 52
 PWRT (PX,PY,CH,NC,IS,IO,IC) ............................................................................. 52
 SET (FL,FR,FB,FT,UL,UR,UB,UT,LL) ................................................................. 53
 SETI (IX,IY) ............................................................................................................. 53
 SETUSV (VN,IV) ...................................................................................................... 54
 VECTOR (PX,PY) ..................................................................................................... 56
 WTSTR (PX,PY,CH,IS,IO,IC) ................................................................................. 56

Conversion Considerations ......................................................................................... 56
 Integer Coordinates .................................................................................................. 56
 Routine POINTS ...................................................................................................... 57
 Option-setting Routines ............................................................................................. 58
 Intensity and Color .................................................................................................... 58
 Spot Size .................................................................................................................. 60
 Condition-access Routines ....................................................................................... 60

August 1987 43
SPPS — An NCAR System Plot Package Simulator

Introduction

The SCD Graphics Project decided some time ago to abandon the NCAR System Plot Package (NSPP) in favor of the Graphical Kernel System (GKS) and has devoted a great deal of time and effort to the transition. That effort is now coming of age; graphics users are now encouraged to switch from the old package to the new.

In the long run, GKS offers important advantages to the user community. In the short run, however, it offers a bit of a headache. GKS is not particularly user-friendly and it is significantly different from NSPP. Therefore, an NSPP look-alike has been created as an interface to GKS. This look-alike is called the System Plot Package Simulator (SPPS). Despite its name, SPPS does not exactly simulate NSPP, nor can it be made to. Some of the major differences between the two are as follows:

- The new routines OPNGKS and CLSGKS may be called to "open" GKS (initialize it) and to "close" GKS (terminate use of it), respectively. By default, these routines specify the use of FORTRAN unit 2 for metacode output. If this use of unit 2 is unacceptable, you may change the metacode unit number with a SETUSV call; you may also elect to do your own calls to open and close GKS and to define the workstations to be used.

- In direct violation of the FORTRAN standard, NSPP allowed x/y coordinate arguments to be expressed either as integers in a plotter coordinate system or as reals in a user coordinate system; SPPS requires that such arguments be expressed as reals.

- The character strings passed to SPPS routines must be in the form of FORTRAN 77 character expressions; Hollerith strings must not be used.

- The routines GETOPT and OPTN have been omitted. The new routines GETUSV and SETUSV provide most of the same capabilities.

- The routines GETCND and SETCND have been omitted; calls to these routines must be modified.

- The obsolete routines AXES, DASHLN, OPTION, PORGN, PScale, PSYM, PWRT, and TICKS have been omitted.

- The fourth argument of the routine POINTS has been redefined in such a way as to avoid violating the FORTRAN standard and to enhance the utility of the routine.

- Functions allowing you to map values from any NSPP/SPPS coordinate system to any other one have been added.

- Two new routines, GETUSV and SETUSV, have been implemented in order to allow you to access "user state variables."

August 1987
The routines GRID, GRIDAL, GRIDL, HALFAX, LABMOD, PERIM, PERIML, and TICK4 are now considered higher-level utilities. As such, they are not a part of SPPS and are not described here. These routines are part of GRIDAL now. See GRIDAL in the "Utilities" section if you need more information.

The flash-package routines FLASH1, FLASH2, FLASH3, and FLASH4 are not in NCAR Graphics. It is hoped that such a facility can be made available in the future by using features of GKS Level 2B.

Coordinate Systems

NSPP and SPPS both use the following coordinate systems:

- **Metacode Coordinates.** The metacode coordinates of a point are integers IMX and IMY between 0 and 32767 (2**15-1), inclusive. The area addressed is a square in a "metacode space" that is usually mapped into a square subset of the addressable area on the plotting device. Metacode coordinates are used in calls to the routine PLOTIT and are returned by the routine FL2INT.

- **Plotter Coordinates.** The plotter coordinates of a point are integers IPX and IPY, where IPX is between 1 and 2**MX and IPY between 1 and 2**MY, inclusive. Note that MX and MY are user state variables, with default values 10 and 10. The values of MX and MY can be set by using the routine SETI and retrieved by using the routine GETSI. Plotter coordinates IPX and IPY are related to metacode coordinates IMX and IMY by the following formulas:
  
  \[
  \text{IMX} = \text{IFIX} \left(32767 \times \text{FLOAT}(\text{IPX}-1)/(2.**\text{MX}-1.)\right) \\
  \text{IMY} = \text{IFIX} \left(32767 \times \text{FLOAT}(\text{IPY}-1)/(2.**\text{MY}-1.)\right)
  \]

Plotter coordinates were used in calls to NSPP routines such as LINE and POINT to specify positions on the plotter frame; they may not be used in calls to the SPPS equivalents of such routines. Plotter coordinates could also be used to specify character widths in calls to the NSPP routine PWRIT; they are still so used in calls to the SPPS routine PWRIT.

- **User Coordinates.** The user coordinates of a point are real numbers RUX and RUY. The relationship between user coordinates and plotter coordinates is determined by the last call to the routine SET, which has the form

  \[
  \text{CALL SET (PL,PR,PB,PT,UL,UR,UB,UT,LL)}
  \]

PL and PR define the left and right edges, and PB and PT the bottom and top edges, of a rectangular subset of the plotter frame. In direct violation of the FORTRAN standard, the NSPP version of SET allowed these to be stated either as real coordinates in the fractional coordinate system, as shown above, or as integer coordinates (call them IL, IR, IB, and IT) in the plotter coordinate system; the SPPS version only allows real values. Default values are 0., 1., 0., and 1., respectively.
UL and UR define the user x-coordinate values to be mapped into the left and right edges, and UB and UT the user y-coordinate values to be mapped into the bottom and top edges, of the area defined by PL, PR, PB, and PT.

LL defines whether the mappings are to be linear or logarithmic, as follows:

1 -- x linear, y linear
2 -- x linear, y logarithmic
3 -- x logarithmic, y linear
4 -- x logarithmic, y logarithmic

Assuming that IL and IR are the plotter-coordinate-system equivalents of PL and PR, the linear mapping of a real user x coordinate RUX into an integer plotter x coordinate IPX is defined as follows:

$$IPX = IL + \text{IFIX} \left( \frac{RUX - UL}{UR - UL} \times \text{FLOAT}(IR - IL) \right)$$

The logarithmic mapping of RUX into IPX is defined as follows:

$$IPX = IL + \text{IFIX} \left( \frac{\text{ALOG10}(RUX) - \text{ALOG10}(UL)}{\text{ALOG10}(UR) - \text{ALOG10}(UL)} \times \text{FLOAT}(IR - IL) \right)$$

The linear and logarithmic mappings of RUY into IPY are similarly defined.

User coordinates are used in calls to routines like POINT and LINE, to specify positions on the plotter frame.

- **Fractional Coordinates.** The fractional coordinates of a point are real numbers RFX and RFY, between 0. and 1., inclusive. An x-coordinate value of 0. indicates the left edge of the usable portion of the plotter frame and the value 1. the right edge of the usable portion of the plotter frame. A y-coordinate value of 0. indicates the bottom edge of the usable portion of the plotter frame and the value 1. indicates the top edge of the usable portion of the plotter frame.

Fractional coordinates RFX and RFY are related to plotter coordinates IPX and IPY by the following formulas:

$$IPX = 1 + \text{IFIX}(RFX \times (2.**MX - 1.))$$
$$IPY = 1 + \text{IFIX}(RFY \times (2.**MY - 1.))$$

Fractional coordinates may only be used as the first four arguments of a call to the routine SET.

**Coordinate-system Mapping Functions**

To facilitate the conversion of user programs from NSPP to SPPS, 24 functions have been provided. Each function accepts a value of a coordinate in one coordinate system and returns a value in another coordinate system. The 24 functions are as follows:
In each case, the first letter of the name stands for "Convert," spelled with a "C" if the value of the function is of type real and with a "K" if the value of the function is of type integer. The second and third letters of the name indicate the coordinate systems of the argument and of the function value, respectively. The fourth letter of the name indicates whether an x coordinate or a y coordinate is being converted. Thus, CPUX(IX) is the real x coordinate in the user system that corresponds to the integer x coordinate IX in the plotter system, and KPMY(IY) is the integer y coordinate in the metacode system that corresponds to the integer y coordinate IY in the plotter system.

SPPS Subroutines and Their Relationship to GKS

The SPPS subroutines are described below. Each description includes a discussion of GKS considerations (what GKS routines the SPPS routine calls and what effect, if any, it has on the state of GKS).

CLSGKS

Closes GKS after output by SPPS.

GKS considerations: CLSGKS calls GDAWK to deactivate metacode workstation 1, GCLWK to close workstation 1, and GCLKS to close GKS.

CURVE (PX, PY, NP)

Draws the curve defined by the points ((PX(I),PY(I)), I=1,NP) in the user coordinate system. The "pen" is left at the point (PX(NP),PY(NP)).

GKS considerations: If there has been no call to SET or if the mapping defined by the last SET call was linear in both x and y, CURVE calls GPL to create a single NP-point polyline; otherwise, the coordinates are transformed 10 at a time to create a set of 10-point polylines whose union is the desired curve.

FLUSH

Flushes the PLOTIT buffer. See the description of PLOTIT, below.

GKS considerations: None.
**FL2INT (PX,PY,IX,IY)**

Given the coordinates, PX and PY, of a point in the user coordinate system, FL2INT returns the coordinates, IX and IY, of that point in the metacode coordinate system. The effect of the statement

```
CALL FL2INT (PX,PY,IX,IY)
```

is the same as that of the two statements

```
IX=KUMX (PX)
IY=KUMY (PY)
```

**GKS considerations:** None.

**FRAME**

Advances to a new frame.

**GKS considerations:** FRAME clears all open and active GKS workstations.

**FRSTPT (PX,PY)**

Generates a pen-up move to the point (PX,PY), in the user coordinate system. Used in conjunction with VECTOR (described below) to draw lines. The statement

```
CALL FRSTPT (PX, PY)
```

is equivalent to the statement

```
CALL PLOTIT (KUMX (PX), KUMY (PY), 0)
```

**GKS considerations:** See PLOTIT.

**GETSET (FL,FR,FB,FT,UL,UR,UB,UT,LL)**

Retrieves the parameters defining the relationship between the user coordinate system and the fractional coordinate system. See the description of SET, below.

Note that the NSPP routine GETSET returned integer values, in the plotter coordinate system, for the first four arguments of GETSET. The SPPS version returns real values, in the fractional system, so as to be consistent with the routine SET.

**GKS considerations:** FL, FR, FB, and FT are simply the values defining the viewport of the current normalization transformation. If there has been no call to SET or if the mapping defined by the last SET call was linear in both x and y, UL, UR, UB, and UT are simply the values defining the window of the current normalization transformation and LL is a 1; otherwise, two or more of the values are of the form 10.**x, where x is a window-defining parameter and LL is not a 1. The order of UL, UR, UB, and UT may be different from that of the values defining the window of the current

August 1987
normalization transformation; see the paragraph describing GKS considerations for the routine SET.

**GETSI (IX, IY)**

Retrieves the parameters defining the size of the plotter and therefore the nature of the plotter coordinate system. Plotter x coordinates are assumed to lie in the range from 1 to \(2^{**IX-1}\), inclusive, and plotter y coordinates are assumed to lie in the range from 1 to \(2^{**IY-1}\), inclusive.

**GKS considerations**: None.

**GETUSV (VN, IV)**

Retrieves the value of one of the "user state variables," an internal parameter controlling some aspect of the behavior of SPPS. VN is the name of the variable whose value is to be retrieved and IV the value to be returned by GETUSV.

See the description of the routine SETUSV, below.

**GKS considerations**: None.

**LINE (X1, Y1, X2, Y2)**

Draws a line from the point (X1, Y1) to the point (X2, Y2), in the user coordinate system. The statement

```
CALL LINE (X1, Y1, X2, Y2)
```

is equivalent to the statements

```
CALL PLOTIT (KUMX(X1), KUMY(Y1), 0)
CALL PLOTIT (KUMX(X2), KUMY(Y2), 1)
```

**GKS considerations**: See PLOTIT.

**MXMY (IX, IY)**

Retrieves the current pen position, in the plotter coordinate system. The pen position is updated correctly by the routines CURVE, FRSTPT, LINE, PLOTIT, POINT, POINTS, and VECTOR. The routine PWRIT leaves the pen at the point (PX, PY) defined by its first two arguments.

**GKS considerations**: None.

**OPNGKS**

Opens GKS for output by SPPS.

**GKS considerations**: OPNGKS calls GOPKS to open GKS, GOPWK to open metacode workstation 1, and GACWK to activate workstation 2.
The metacode workstation is connected to FORTRAN unit 2; obviously this may conflict with your use of FORTRAN unit 2. If so, use a "CALL SETUSV ('MU', n)" to change to some other unit n.

**PLOTIF (FX,FY,IP)**

Moves the pen to the position (FX,FY) in the fractional coordinate system, with the pen up if IP equals 0 and with the pen down if IP equals 1. For the sake of efficiency, pen moves are buffered. Up to fifty pen moves may be saved before the buffer is flushed; a buffer flush can be forced by a PLOTIF call with IP=2. The effective size of the buffer may be changed by a call to SETUSV:

```
CALL SETUSV ('PB', IS)
```

where IS is an integer between 2 and 50; the value 2 effectively turns buffering off, forcing a buffer flush after every pen move. PLOTIF and PLOTIT use the same pen-move buffer.

**GKS considerations:** Each time PLOTIF's buffer is flushed, 0 or more polylines are created. Because of the buffering, objects created by a mixture of calls to SPPS routines and GKS routines may be drawn in the wrong order. This may be bothersome on a pen-oriented device.

**PLOTIT (IX,IY,IP)**

Moves the pen to the position (IX,IY) in the metacode coordinate system, with the pen up if IP equals 0 and with the pen down if IP equals 1. For the sake of efficiency, calls to PLOTIT are buffered. Up to fifty pen moves may be saved before the buffer is flushed; a buffer flush can be forced by a PLOTIT call with IX=IY=IP=0 or with IP=2. The effective size of the buffer may be changed by a call to SETUSV:

```
CALL SETUSV ('PB', IS)
```

where IS is an integer between 2 and 50; the value 2 effectively turns buffering off, forcing a buffer flush after every pen move. PLOTIT and PLOTIT use the same pen-move buffer.

**GKS considerations:** Each time PLOTIT's buffer is flushed, 0 or more polylines are created. Because of the buffering, objects created by a mixture of calls to SPPS routines and GKS routines may be drawn in the wrong order. This may be bothersome on a pen-oriented device.

**POINT (PX,PY)**

Draws a point at the point (PX,PY), in the user coordinate system. The statement

```
CALL POINT (PX,PY)
```
NCAR Graphics User's Guide

is equivalent to the statements

```
CALL PLOTIT (KUMX(PX), KUMY(PY), 0)
CALL PLOTIT (KUMX(PX), KUMY(PY), 1)
```

**GKS considerations:** None.

**POINTS (PX, PY, NP, IC, IL)**

Marks the points with coordinates (PX(I), PY(I)) for I from 1 to NP. If IC is negative, GKS polymarkers of type -IC are used to mark the points. If IC is 0, GKS polymarkers of type 1 (dots) are used. If IC is greater than 0, the character defined by CHAR(IC) (a FORTRAN-77 function whose value is the character corresponding to the integer value IC) to mark each of the points. If IL is non-zero, a curve is also drawn through the points.

Using the subroutine POINTS causes PLOTIT's buffer to be flushed. The pen is left at the position (PX(NP), PY(NP)).

**GKS considerations:** If there has been no call to SET or if the last call to SET specified a linear mapping in both x and y and if the value of IC specifies the use of a GKS polymarker, a single polymarker is created by a call to GPM and, if IL is non-zero, a single polyline is created by a call to GPL. In all other cases, a more complicated set of graphical objects is created, the union of which has the desired effect.

If IC is greater than 0, calls to GTX are used to draw the desired character at each of the points; the horizontal and vertical text-alignment parameters are altered to center each character on its point and then returned to their initial values. You can alter other text attributes (character orientation, color, etc.) prior to calling POINTS.

**PWRIT (PX, PY, CH, NC, IS, IO, IC)**

This routine is essentially an old version of WTSTR. Like WTSTR, PWRIT draws a character string at a specified position, with a specified size, orientation, and centering, but it has an argument (NC), requiring one to specify the number of characters to be written from CH. (WTSTR uses the FORTRAN 77 function LEN(CH) to figure out for itself how many characters to write.) The statement

```
CALL PWRIT (PX, PY, CH, NC, IS, IO, IC)
```

is equivalent to the statement

```
CALL WTSTR (PX, PY, CH(1:NC), IS, IO, IC)
```

and, in fact, PWRIT executes the latter statement.

Because having to count characters is annoying, error-prone, and, in a FORTRAN 77 environment, unnecessary, one should use WTSTR instead of PWRIT.
GKS considerations: The GKS parameters controlling character height, text path, orientation, and alignment are temporarily altered so that a GTX call can be used to draw the character string; following the GTX call, these parameters are returned to their original values.

**SET (FL, FR, FB, FT, UL, UR, UB, UT, LL)**

Allows you to set the internal parameters defining the mapping from the fractional system to the user system and vice-versa.

FL, FR, FB, and FT are real values, between 0. and 1., inclusive, defining the left, right, bottom, and top edges of a rectangle in the fractional coordinate system.

UL, UR, UB, and UT are real values defining the left, right, bottom, and top edges of a rectangle in the user coordinate system.

LL specifies the linear/log nature of the mapping from one system to the other, as follows:

1. Linear, y linear
2. Linear, y logarithmic
3. Logarithmic, y linear
4. Logarithmic, y logarithmic

The linear mapping of a user x coordinate UX to a fractional x coordinate FX is defined as follows:

\[ FX = FL + (UX - UL) / (UR - UL) \times (FR - FL) \]

The logarithmic mapping of UX to FX is defined as follows:

\[ FX = FL + (\text{ALOG10} (UX) - \text{ALOG10} (UL)) / (\text{ALOG10} (UR) - \text{ALOG10} (UL)) \times (FR - FL) \]

The linear and logarithmic mappings of a user y coordinate UY to a fractional y coordinate FY are defined analogously.

Note that the NSPP routine SET allowed you to use either integers or reals for the first four arguments. The SPPS routine SET requires that the first four arguments be reals.

GKS considerations: Calling SET redefines normalization transformation 1 and makes it the current normalization transformation. The first four arguments define the viewport. The second four arguments (or their base-10 logarithms, depending on the value of LL) define the window. Because of the GKS restriction that the numerical values at the left and bottom edges of the window must be less than the numerical values at the right and top edges, respectively, the user values are swapped as necessary, and the internal "user state variable" identified by 'MI' is used to remember which values have been swapped.

**SETI (IX, IY)**

Allows you to set the parameters defining the size of the plotter and therefore, the nature of the plotter coordinate system. Plotter x coordinates are...
assumed to lie in the range from 1 to \(2^{\text{IX}-1}\), inclusive, and plotter y coordinates are assumed to lie in the range from 1 to \(2^{\text{IY}-1}\), inclusive.

**GKS considerations:** None.

**SETUSV (VN,IV)**

Sets the value of one of the "user state variables," an internal parameter controlling some aspect of the behavior of SPPS. VN is the name of the variable whose value is to be set, and IV is the value which is to be given to it.

**Table 3 — User-state Variables**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Possible Values</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>'LS'</td>
<td>1, 2, 3, or 4</td>
<td>Linear-log scaling</td>
</tr>
<tr>
<td>'MI'</td>
<td>1, 2, 3, or 4</td>
<td>Mirror-imaging flag</td>
</tr>
<tr>
<td>'XF'</td>
<td>1, 2, ..., 15</td>
<td>X axis scale factor</td>
</tr>
<tr>
<td>'YF'</td>
<td>1, 2, ..., 15</td>
<td>Y axis scale factor</td>
</tr>
<tr>
<td>'PB'</td>
<td>2, 3, ..., 50</td>
<td>PLOTIT buffer size</td>
</tr>
<tr>
<td>'MU'</td>
<td>a unit number</td>
<td>Metacode output unit</td>
</tr>
<tr>
<td>'IR'</td>
<td>1, 2, ...</td>
<td>Red intensity</td>
</tr>
<tr>
<td>'IG'</td>
<td>1, 2, ...</td>
<td>Green intensity</td>
</tr>
<tr>
<td>'IB'</td>
<td>1, 2, ...</td>
<td>Blue intensity</td>
</tr>
<tr>
<td>'IN'</td>
<td>1, 2, ..., 10000</td>
<td>Overall intensity</td>
</tr>
<tr>
<td>'IM'</td>
<td>1, 2, ...</td>
<td>Maximum color index</td>
</tr>
<tr>
<td>'II'</td>
<td>1, 2, ..., IM</td>
<td>Restore color index</td>
</tr>
<tr>
<td>'LW'</td>
<td>1, 2, ...</td>
<td>Line width</td>
</tr>
<tr>
<td>'MS'</td>
<td>1, 2, ...</td>
<td>Marker size</td>
</tr>
</tbody>
</table>

The variable referenced by 'LS' is the same as the one referenced by the argument LL in a call to SET or GETSET. The variable referenced by 'MI' has similar values and indicates whether or not the mappings of user x and y coordinates are to be normal or mirrored (reversed end-for-end). Both 'LS' and 'MI' are normally set by a call to the routine SET, rather than by a call to SETUSV.

The variables referenced by 'XF' and 'YF' are the same as those referenced by the arguments IX and IY in a call to GETSI or SETI.

The variable 'MU' specifies the number of the unit to which metacode is to be written.

All of the variables 'IR', 'IG', 'IB', 'IN', 'IM', and 'II' have to do with setting intensity and color. See the section "Intensity and Color" below, for a complete discussion of these variables. Assuming that IR, IG, IB, and IN are the variables referenced by the names 'IR', 'IG', 'IB', and 'IN', respectively, the red, green, and blue intensities are set as follows:
The variables 'IR', 'IG', and 'IB' must have positive values. They determine how the desired color is to be mixed (additively) from red, green, and blue primaries; the default values are 1, 1, and 1, giving white. The variable 'IN' must have a value between 0 and 10000; it specifies the desired intensity, in ten-thousandths of the maximum value; its default value is 8000. The GKS calls actually setting the color and intensity are performed during the call setting 'IN'; therefore, to set both the color and the intensity, set any or all of 'IR', 'IG', and 'IB', and finally, 'IN'. Because of the way in which the GKS metacode translator works when driving a device with only two possible intensities (on and off) values of 'IN' less than 5000 should be avoided. On the DICOMED, this is reasonable anyway, because intensities in the bottom half of the scale are very dim.

The variable 'IM' specifies the maximum color index to be used by SPPS; the value given to it must be appropriate for the device being driven. The default value is 1, which is appropriate for the DICOMED.

The variable 'II' specifies the current color index; the value given should be one previously retrieved by a call to GETUSV.

The variables 'LW' and 'MS' specify the line width and marker size, respectively. Each is stated in thousandths of the normal value. For example, the value 2000 represents twice the default line width or marker size. The default value of both parameters is therefore 1000.

GKS considerations: Depends on the value of the first argument, as follows:

'MU' (Metacode Unit) If OPNGKS has been called, the workstation with workstation identifier 1 is deactivated and closed and a new workstation with workstation identifier 1 is opened and activated. If OPNGKS has not been called, there is no effect on GKS.

'IN' (INtensity) The aspect source flags for all color indices are set to "individual;" a new color index is assigned for use with polylines, polymarkers, text, and fill-areas; and the color associated with that color index is redefined for each open workstation.

'II' (Intensity Index) The aspect source flags for all color indices are set to "individual;" and the color index specified by the second argument is assigned for use with polylines, polymarkers, text, and fill-areas.

'LW' (Line Width) The aspect source flag for line width is set to "individual;" and the line width scale factor is reset.

'MS' (Marker Size) The aspect source flag for the marker size scale factor is set to "individual;" and the marker size scale factor is reset.
VECTOR (PX,PY)

Generates a pen-down move to the point (PX,PY), in the user coordinate system. Used in conjunction with FRSTPT (described above) to draw lines. The statement

CALL VECTOR (PX, PY)

is equivalent to the statement

CALL PLOTIT (KUMX (PX), KUMY (PY), 1)

GKS considerations: See PLOTIT.

WTSTR (PX,PY,CH,IS,IO,IC)

Draws the character string defined by the character expression CH, at the position (PX,PY), in the user coordinate system.

IS is the desired character width in the plotter coordinate system; the value 0 is interpreted to mean 8, the value 1 to mean 12, the value 2 to mean 16, the value 3 to mean 24, and values greater than or equal to 4 as themselves.

IO is the desired orientation of the string, in degrees counterclockwise from horizontal; 0 means that the string will be written from left to right on the plotter frame; 90 means that it will be written from bottom to top, right side up as viewed from the right side of the frame, etc.

IC is the centering option. If IC is negative, (PX,PY) will be in the center of the left edge of the leftmost character. If IC is 0, (PX,PY) will be in the center of the whole string. If IC is greater than 0, (PX,PY) will be in the center of the right edge of the rightmost character.

The PLOTIT buffer is flushed before the character string is drawn. The pen is left at the point (PX,PY).

GKS considerations: The GKS parameters controlling character height, text path, orientation, and alignment are temporarily altered so that a GTX call can be used to draw the character string; following the GTX call, these parameters are returned to their original values.

Conversion Considerations

Converting from NSPP to SPPS involves certain changes for the user.

Integer Coordinates

In violation of the FORTRAN standard, NSPP drawing routines allowed coordinates to be given as integers in the plotter coordinate system or as reals in the user coordinate system. SPPS allows only the latter. The functions CPUX and CPUY may be used to map values from the plotter coordinate system to the user coordinate system. For example, the NSPP
CALL POINT (IX, IY)
can be rewritten as the SPPS statement
CALL POINT (CPUX(IX), CPUY(IY))
This technique can be used for calls to the routines FRSTPT, LINE, POINT, PWRIT, and VECTOR (and for calls to GRIDAL and HALFAx, as well). Calls to the NSPP routines CURVE and POINTS are not so easily modified. Their SPPS equivalents will only accept real user coordinates in the input arrays; user code which called these routines with integer coordinates will have to be changed.

Note that this problem does not arise with the routine PLOTIT, which, in both NSPP and SPPS, is called with integer coordinates in the metacode system.

The first four arguments of the NSPP routine SET could be given either as integers in the plotter coordinate system or as reals in the fractional coordinate system. Calls to the SPPS routine SET must use reals. The functions CPFX and CPFY may be used to do the required conversion. For example, the NSPP statement
CALL SET (IL, IR, IB, IT,....)
becomes the SPPS statement
CALL SET (CPFX(IL), CPFX(IR), CPFY(IB), CPFY(IT),....)
The NSPP routine GETSET returned integers in the plotter coordinate system for its first four arguments; the SPPS equivalent returns reals in the fractional system. Thus, the NSPP statement
CALL GETSET (IL, IR, IB, IT,....)
may be replaced by the SPPS statements
CALL GETSET (VL, VR, VB, VT,....)
IL=KFPX(VL)
IR=KFPX(VR)
IB=KFPY(VB)
IT=KFPY(VT)

Routine POINTS

The fourth argument of POINTS has been redefined. The NSPP call
CALL POINTS (PX, PY, NP, 0, IL)
may be left unchanged, but the NSPP call
CALL POINTS (PX, PY, NP, '+', IL)
must be changed. The simplest SPPS equivalent is
CALL POINTS (PX, PY, NP, ICHAR('+'), IL)
which will still give a plus sign at each point. It is more efficient, however, to use GKS polymarkers when possible. The SPPS call
CALL POINTS (PX, PY, NP, -2, IL)
uses the plus sign polymarker. The value -1 indicates dots, the value -2 plus signs, the value -3 asterisks, the value -4 circles, and the value -5 diagonal crosses.

Option-setting Routines

The NSPP routines OPTN and GETOPT have not been implemented in SPPS, but the routines GETUSV and SETUSV implement many of the same features.

Intensity and Color

The NSPP call

CALL OPTN ('IN', IN)

allowed you to set the intensity with which lines, points, and text were subsequently to be drawn. The value of IN was required to lie between 0 and 255. This statement may be replaced by

CALL SETUSV ('IN', IFIX(10000.*FLOAT(IN)/255.))

which is almost identical, except that the second argument is between 0 and 10000.

On a device with color capability, you need a mechanism to select the color to be used for subsequent lines, points, and text. With NSPP, this was done rather clumsily, using calls like this:

CALL OPTN ('CO', 'RED')

Only a few colors — typically 'WHITE', 'RED', 'BLUE', 'GREEN', 'CYAN', 'MAGENTA', and 'YELLOW' — were available. SPPS provides a way to get a desired color by mixing red, green, and blue light. The call above would be replaced by the SPPS calls

CALL SETUSV ('IR', 1)
CALL SETUSV ('IG', 0)
CALL SETUSV ('IB', 0)
CALL SETUSV ('IN', IN)

which says that the desired color is to be blended using 1 part red light to 0 parts green light to 0 parts blue light and that the intensity is to be IN/10000 times the maximum possible. The RGB ratio 0:1:0 forms green, 0:0:1 blue, 1:1:0 yellow, 1:0:1 magenta, 0:1:1 cyan, and 1:1:1 white. The default is 1:1:1 at intensity 8000/10000 (white light at .8 maximum). Other RGB ratios form other colors. In general, you must have an RGB color cube available in order to select a meaningful RGB ratio. (This is the one drawback to the RGB scheme, which was chosen because it is used by GKS. A scheme based on hue, saturation, and intensity is more user-friendly; such a scheme could be built on top of the current one.)

It is important to realize that, in the GKS scheme, the setting of color and intensity are inseparable and that the GKS calls setting them occur as a result of the fourth call in the above sequence — the one setting 'IN'.
Resetting one or more of 'IR', 'IG', and 'IB' will have no effect until after the next call setting 'IN'.

Devices like a DICOMED or a pen plotter record lines, points, and text serially. If, on such a device, you set the color to red and draw a line and then reset the color to blue and draw another line, you can expect to get the obvious — a red line and a blue line. On devices having a screen that is continuously refreshed, the procedure is more complicated. Typically, each object drawn on such a device is associated with a "color index" — a pointer into a "color table," each element of which defines a particular color and intensity. If an object is drawn with color index 1 and then color table entry 1 is redefined, and then a second object is drawn with color index 1, the two objects will differ in color only until the next time the screen is refreshed — a fraction of a second. Thus, a different color index must be used for each desired color; the problem with that, of course, is that the number of such color indices is limited.

By default, SPPS uses only color index 1, which is appropriate for the DICOMED or for a pen plotter. The SPPS call

\[ \text{CALL SETUSV ('IM', IM)} \]

where IM is a positive integer, tells SPPS to use color indices 1 through IM. Such a call should be done once, at the beginning of the program; the value of IM should be appropriate to the graphics device being used. Subsequently, each time the intensity/color is changed, SPPS updates its current color index \( \text{II} \) to have the new value \( \text{MOD(II,IM)+1} \) and uses that value for drawing subsequent objects. Note that \( \text{II} \) cycles repeatedly through the color indices 1 through IM.

SPPS has no mechanism allowing it to detect when the same intensity/color is being reused. Consider, for example, what happens if you set \( \text{'IM'} \) to 3 and draw four objects — the first and third in red, and the second and fourth in blue. If you did this in the obvious way — by changing the color before drawing each new object — SPPS would use color index 1 for red, 2 for blue, 3 for red, and, finally, 1, again, for blue. At that point, the color of object 1 would change from red to blue. To avoid this problem, you can recover from SPPS the value of the color index used for a particular intensity/color and then re-use that value as needed. The SPPS calls

\[
\begin{align*}
\text{CALL SETUSV ('IR', 1)} \\
\text{CALL SETUSV ('IG', 0)} \\
\text{CALL SETUSV ('IB', 0)} \\
\text{CALL SETUSV ('IN', 5000)} \\
\text{CALL GETUSV ('II', I1)} \\
\end{align*}
\]

define a half-intensity red and retrieve the color index associated with it in \( \text{II} \). Thereafter, you can use the SPPS call

\[ \text{CALL SETUSV ('II', I1)} \]

to request that the same half-intensity red be used. If necessary, then, you can define, up front, all the colors/intensities you are going to use, save the associated color indices in an array, and use SPPS calls like the one above to switch from one color/intensity to the next.
Spot Size

The NSPP call

```c
CALL OPTN ('SS',16)
```

sets the "spot size" (i.e. the thickness of the beam or pen used to draw objects) to twice its default value of 8. The value provided was a value in the metacode coordinate system — 16 really meant 16/32767ths of the distance across the screen. These values were appropriate for the DICOMED.

The SPPS call

```c
CALL SETUSV ('LW', 2000)
```

would have the same effect; it requests the use of a beam/pen twice the normal width for the device you are using. The effect of this call is seen only in lines drawn; characters (generated by calls to PWRIT, WTSTR, or POINTS) and markers (generated by calls to POINTS) are unaffected.

Condition-access Routines

The NSPP routines SETCND and GETCND do not have SPPS counterparts.

The NSPP call

```c
CALL SETCND ('XT',n)
```

in which n had the value 1, 2, or 3, was intended to tell the plot package whether x coordinates in calls to routines such as POINT and LINE were to be integers or reals (n=1), integers only (n=2), or reals only (n=3). Since the SPPS equivalents require that such x coordinates be reals, the above call has no equivalent in SPPS. The NSPP call

```c
CALL SETCND ('YT',n)
```

is analogous, but applied to y coordinates. The NSPP calls

```c
CALL GETCND ('MU',n)
CALL SETCND ('MU',n)
```

were intended to get/set the current metacode output unit number. They may be replaced by the SPPS calls

```c
CALL GETUSV ('MU',n)
CALL SETUSV ('MU',n)
```

It is unclear at the time of writing what the effect of switching to a previously written metacode unit will be. It is possible that the new metacode output will simply overwrite the old. Check with your NCAR Graphics site representative to determine the current situation.
# Section IV: NCAR Graphics Utilities, GKS Level 0A

## Introduction

Changes from the Previous Version

Graphics Utilities Categorized by Functionality

Brief Descriptions of the Graphics Utilities

Detailed Descriptions of Each Graphics Utility

<table>
<thead>
<tr>
<th>Utility</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREAS</td>
<td>69</td>
</tr>
<tr>
<td>AUTOGRAPH</td>
<td>89</td>
</tr>
<tr>
<td>CONRAN</td>
<td>105</td>
</tr>
<tr>
<td>CONRAQ</td>
<td>123</td>
</tr>
<tr>
<td>CONRAS</td>
<td>137</td>
</tr>
<tr>
<td>CONREC</td>
<td>155</td>
</tr>
<tr>
<td>CONRECQCK</td>
<td>163</td>
</tr>
<tr>
<td>CONRECSPR</td>
<td>169</td>
</tr>
<tr>
<td>DASHCHAR</td>
<td>177</td>
</tr>
<tr>
<td>DASHLINE</td>
<td>183</td>
</tr>
<tr>
<td>DASHSMTH</td>
<td>187</td>
</tr>
<tr>
<td>DASHSUPR</td>
<td>193</td>
</tr>
<tr>
<td>EZMAP</td>
<td>201</td>
</tr>
<tr>
<td>EZMAPA</td>
<td>287</td>
</tr>
<tr>
<td>GRIDAL</td>
<td>303</td>
</tr>
<tr>
<td>HAFTON</td>
<td>313</td>
</tr>
<tr>
<td>HISTGR</td>
<td>321</td>
</tr>
<tr>
<td>ISOSRF</td>
<td>333</td>
</tr>
<tr>
<td>ISOSRFHR</td>
<td>339</td>
</tr>
<tr>
<td>PWRITX</td>
<td>345</td>
</tr>
<tr>
<td>PWRITY</td>
<td>353</td>
</tr>
<tr>
<td>PWRZI</td>
<td>357</td>
</tr>
<tr>
<td>PWRZS</td>
<td>361</td>
</tr>
<tr>
<td>PWRZT</td>
<td>365</td>
</tr>
<tr>
<td>SRFACE</td>
<td>369</td>
</tr>
<tr>
<td>STRMLN</td>
<td>377</td>
</tr>
<tr>
<td>THREED</td>
<td>385</td>
</tr>
<tr>
<td>VELVCT</td>
<td>391</td>
</tr>
</tbody>
</table>
NCAR Graphics Utilities, GKS Level 0A

Introduction

To help you find the utilities you need to use, this introduction groups the utilities in two ways:

- by functionality.
- alphabetically.

The full description of each utility follows this introduction. The utility descriptions are organized alphabetically. The utility name appears in the upper right corner of right-hand pages to help you locate utilities when thumbing through the book. You may want to add extra dividers in front of the utilities you refer to most frequently.

There are examples of EZMAP code and output in this section; see the "Examples" section for examples of the other utilities.

Changes from the Previous Version

The utilities descriptions are identical to those in The NCAR GKS-Compatible Graphics System, (NCAR/TN-267+IA, April 1986), with the following exceptions:

- The utilities AREAS and EZMAPA have been added.
- The former utility HSTGRM has been rewritten to correct errors and significant new features have been added. Because of these changes, the old name was dropped and the new utility is called HISTGR.
- Two changes were made in EZMAP. MAPEOS was replaced with MAPEOD, which has different arguments, and the seven intensity parameters were replaced by color-index parameters. The EZMAP example codes were rewritten for execution in a GKS environment. Consult the EZMAP documentation for details.
- A shortened description of AUTOGRAPH was added to this section. AUTOGRAPH: A Graphing Utility (NCAR/TN-245+IA) has been updated and is available as a separate manual, which was shipped to you along with this manual. The example codes were rewritten for execution in a GKS environment.
- Many of the utility descriptions have been updated to incorporate minor changes.

Graphics Utilities Categorized by Functionality

If you have a particular graphics function in mind, but don't know the name of the appropriate graphics utility, use Table 4 to locate the correct utility.
Table 4 — Utilities by Function

| Contoured lines       | CONRAN   |
|                      | CONRAQ   |
|                      | CONRAS   |
|                      | CONREC   |
|                      | CONRECSPR|
|                      | CONRECQCK|
| Dashed lines         | DASHCHAR |
|                      | DASHLINE |
|                      | DASHSMTH |
|                      | DASHSUPR |
| Map outlines         | EZMAP    |
| Solid-colored maps   | EZMAPA   |
| Graphs               | AUTOGRAPH|
|                      | HISTGR   |
| Graph backgrounds    | GRIDAL   |
| Halftones (gray scale)| HAFTON  |
| Character generation | PWRITX   |
|                      | PWRITY   |
|                      | PWRZI    |
|                      | PWRZS    |
|                      | PWRZT    |
| Defining areas       | AREAS    |
| Streamlines          | STRMLN   |
| Three-dimensional displays | ISOSRF |
|                      | ISOSRFHRI|
|                      | SRFACE   |
|                      | THREED   |
| Vectors              | VELVCT   |

**Brief Descriptions of the Graphics Utilities**

A brief overview of each utility follows, along with the subroutines and utilities it requires. The utilities are listed here by the 8-character-or-less name that is used on the NCAR Graphics distribution tape. Three of the utilities are most commonly referred to by longer names — AUTOGRAPH (for...
Utilities

AUTOGRPH), CONRECQCK (for CONRCQCK), and CONRECSPR (for
CONRCSPR).

AREAS: Creates an area map from a set of edges that divide a two-
 dimensional plane into areas. The area map may then be used in a number
of ways, such as in conjunction with EZMAPA to create solid-color maps.

AUTOGRPH: Draws and annotates curves or families of curves. Requires
DASHCHAR (or DASHSMTH, if smoothed curves are desired). This utility
is also referred to as AUTOGRAPH.

CONRAN: Contours irregularly spaced data, labeling the contour lines.
Requires DASHCHAR (or DASHSMTH, if smoothed curves are desired).
Also requires CONCOM and CONTERP.

CONRAQ: Like CONRAN, but smaller and faster because it has no label-
ing capacity. Also requires CONTERP.

CONRAS: Like CONRAN, but bigger and slower because lines are
smoothed and crowded lines are removed. Also requires CONCOM and
CONTERP.

CONREC: Contours two-dimensional arrays, labeling the contour lines.
Requires DASHCHAR (or DASHSMTH, if smoothed contour lines are
desired).

CONRCQCK: Like CONREC, but faster and smaller because contours
are unlabeled. This package shares entry names with CONREC, so they
cannot both be included in a binary library. This utility is also referred to
as CONRECQCK.

CONRCSPR: Like CONREC, but bigger and slower because contours are
smoothed and labeled, and crowded lines are removed. This package shares
entry names with CONREC, so they cannot both be included in a binary
library. This utility is also referred to as CONRCSPR.

DASHCHAR: Software dashed-line package with labeling capability.

DASHLINE: Like DASHCHAR, but smaller and faster because it has no
labeling capacity. This package shares entry names with DASHCHAR, so
they cannot both be included in a binary library.

DASHSMTH: Like DASHCHAR, but bigger and slower because lines are
smoothed. This package shares entry names with DASHCHAR, so they can-
ot both be included in a binary library.

DASHSUPR: Like DASHCHAR, but bigger and slower because lines are
smoothed and crowded lines are removed. This package shares entry names
with DASHCHAR, so they cannot both be included in a binary library.
EZMAP: Plots continental, U.S. state, and/or world political outlines according to one of ten projections. Requires the continental outline data base EZMAPDAT.

EZMAPA: Allows EZMAP output to be redirected to routines in the AREAS package. EZMAPA makes it possible to create solid-color world maps and to draw lines on a map that are masked by the areas created by the area map. For example, latitude and longitude lines can be drawn over water and omitted over land masses. Requires use of AREAS and EZMAP.

GRIDAL: Package for drawing graph paper, backgrounds, perimeters, and so on.

HAFTON: Halftone (gray scale) pictures from a two-dimensional array.

HISTGR: General purpose package for drawing histograms (bar charts).

ISOSRF: Iso-surfaces (with hidden lines removed) from a three-dimensional array. May optionally include PWRZI for drawing characters in three space.

ISOSRFHR: Iso-surfaces (with hidden lines removed) from a high resolution three-dimensional array.

PWRITX: Draws fancy characters, using the Hershey database. Requires that the data base files PWRITXC1, PWRITXC2, PWRITXD1, and PWRITXD2 be run first through the program PWRITXNT to turn card-image fonts into binary form.

PWRITY: Draws simple software characters.

PWRZI: To be used with ISOSRF to draw characters in three space.

PWRZS: To be used with SRFACE to draw characters in three space.

PWRZT: To be used with THREED to draw characters in three space.

SRFACE: Three-dimensional display of a function of two variables (with hidden lines removed). May optionally include PWRZS for drawing characters in three space.

STRMLN: Plots a representation of any two-dimensional vector field for which planar vector components are given on a regular rectangular lattice, displaying both field direction (via lines of flow containing arrowheads and feathers) and field magnitude (based on distance between those flow lines).

THREED: Provides three-space line drawing capabilities, with entry points equivalent to the line drawing entry points of the System Plot Package Simulator. May optionally include PWRZT for drawing characters in three space.
VELVCT: Two-dimensional velocity field displayed by drawing arrows from the data locations.

Detailed Descriptions of Each Graphics Utility

The detailed descriptions for using each utility are on the following pages.
AREAS
AREAS

BACKGROUND

Several of the NCAR graphics utilities produce lines that divide the plane into areas of interest. For example, the contouring packages CONREC and CONRAN draw contour lines that divide the plane into contour bands and the map-drawing utility EZMAP draws boundary lines that divide the plane into areas that represent continents, oceans, islands, lakes, and countries. With the advent of GKS and of hardware devices with color-fill capability, it has become important to obtain the definitions of the areas defined by a set of edges so that they may be color-filled, shaded, or used as masks while drawing other lines. The package AREAS provides a general way to do this.

In AREAS, the lines that divide the plane are called edges. Given a set of edges that divide a two-dimensional plane into areas, AREAS creates a data structure called an area map. The area map can then be used to color-fill or shade areas and mask lines.

The terms edge, area, area map, and mask are defined fully below, along with other key terms you need to understand to use AREAS effectively.

DOCUMENT ORGANIZATION

The rest of this document on AREAS contains the following sections, in this order:

- **Terms and Concepts:** Definitions and illustrations of the key terms used in AREAS.
- **Current Status:** Current and future planned uses of AREAS.
- **Required Steps to Prepare Area Maps:** Brief descriptions of the three routines you must use in preparing an area map.
- **Uses of an Area Map:** Brief descriptions of the three routines used to obtain definitions of areas, mask lines, and obtain area identifiers associated with a given point.
- **Details of Routines:** Complete details of the six routines discussed in the two previous sections. Routines are in alphabetical order.

TERMS AND CONCEPTS

Basic terms and concepts that you need to understand are defined here.

**User Coordinate System**

The *user coordinate system* is defined by the last call to the SPPS routine SET or by the equivalent calls to GKS routines. Such a call specifies the minimum and maximum values of user-system X's and Y's and how such values are to be mapped into the *fractional coordinate system*. You define edges for AREAS using coordinates in the user coordinate system.

**Fractional Coordinate System**

The *fractional coordinate system* is one in which X and Y coordinates take on values between 0. and 1., inclusive. Fractional coordinates are identical to *normalized device coordinates* in GKS. Areas are defined using coordinates in the fractional coordinate system.

August 1987
**Edges**

An *edge* is a sequence of straight-line segments, joined end-to-end. Each edge is defined by: 1) an ordered set of two or more points that you specify in the user coordinate system, 2) an integer identifying the group to which the edge belongs, 3) another integer identifying what area is to the left of the edge, and 4) another integer identifying what area is to the right of the edge.

Most useful groups of edges will obey the following rules:

- the line segments that make up one edge should not cross:

  This: 

  ![](image1)

  Not this:

  ![](image2)

- There should only be one area to the left and one area to the right of an edge:

  This: 

  ![](image3)

  Not this (The edge defined by points 1, 2, and 3 has 2 areas to its right.):

  ![](image4)

**Group of Edges**

A *group of edges* is a set of edges that were generated by a common source. A group of edges divides the plane into areas that are useful to you. For example, one group of edges might be the set of all lines produced by projecting the U.S. state boundary lines. The set of all lines produced by contouring a temperature field might constitute a second group of edges.
A group of edges:  

Another group of edges:

**Group Identifier**  
A *group identifier* is a positive, non-zero integer (1, 2, ...) that is associated with a particular group of edges.

**Area**  
An *area* is one of the polygonal regions of the plane created by the edges in one or more groups of edges. For example, the edges obtained by projecting the U.S. state boundary lines produce as areas the states (plus islands that are parts of states, lakes that are included in states, and the like). Adding another group of edges obtained by projecting temperature contours creates a more complicated set of areas, each of which must be characterized by a geographical location and a temperature range. An area created by the AREAS package always has associated with it as many *area identifiers* (see definition below) as there are groups of edges in the area map from which the area came.

Areas created by two groups of edges:

**Area Map**  
An *area map* is a linked list of information that is constructed by calls to routines in the AREAS package, in an array that you provide. The area map contains all of the relevant information about one or more groups of edges. The definitions of areas created by the edges can be obtained from the area map.
**Perimeter**

The *perimeter* of a group of edges is a subgroup of the group. Each subgroup member has either a left or a right area identifier that is negative, specifying that what is to that side of the edge is outside the region of interest for the group. Typically, the other area identifier for the edge has a zero value.

A *default perimeter* is automatically supplied for each group of edges. The default perimeter consists of the largest possible square, traced in a clockwise direction, with a *left area identifier* of -1 and a *right area identifier* of 0. (The terms *left area identifier* and *right area identifier* are defined below.)

Default perimeter:

![Default Perimeter Diagram](attachment:image.png)

You may specify a perimeter for each group of edges; areas outside the perimeter are considered not to be of interest. For example, if land/water boundary lines are projected using an orthographic projection, a circular perimeter defines the edge of visibility; areas outside the perimeter are invisible. The perimeter need not be defined by a simple closed curve; it may be a more complicated topological object having one or more holes.

Below is an example of a doughnut-shaped perimeter; it represents a stereographic projection of that part of the globe between 10° N and 60° N.
Left and Right Area Identifiers

Left and right are defined from the position of the first point of an edge, looking toward the second point of the edge. From such a vantage point, the area to the left of the edge is identified by a left area identifier and the area to the right of the edge is identified by a right area identifier.

Areas to the right and left of the edge defined by the points 1, 2, and 3 are marked as such here:

Area Identifiers

An area identifier is an integer (..., -1, 0, 1, 2, ...) that is associated with an area created by the edges in a single group of edges. As each edge is defined, you must supply area identifiers for the areas to the left and right of the edge. The following types of values may be used:

- A negative value identifies an area that is known to be outside the perimeter of the group.
- A positive value identifies an area that is inside the perimeter of the group.
- A zero value implies an unidentified area. A zero value is used as a left or right area identifier when it is unknown or unimportant what area is to the left or to the right of the edge. This may be because the area identifier information will not be required or because that information will be supplied along with the definition of some other edge of the same area.

Mask

The set of areas defined by an area map can be used as a mask for a line that is being drawn. The line is broken into pieces, each of which lies entirely within one and only one area. Each such piece can be drawn a different way.

Latitude and longitude lines masked over land masses:
AREAS was first introduced in 1987 in response to a perceived need for color-fill capabilities. As of July 1987, EZMAP is the only utility that has been extended to use AREAS. The new routines form a new utility called EZMAPA. These routines can be used to create color-filled global maps. See the EZMAPA documentation for usage details.

Work is in progress to create a new contouring package that will use AREAS to create color-filled contour plots. Other uses of AREAS are expected to evolve in response to user needs.

Since AREAS is a new utility, there may be changes in the existing package. Consult your NCAR Graphics site representative to make sure you are using the latest version.

Detailed descriptions of each of the routines in the package AREAS are given in "Details of Routines" later in this section. The steps listed here indicate the order in which the routines should be called and the basic function of each.

Assume that you have one or more groups of edges, each of which divides the user plane into areas. Remember that each edge is defined by an ordered set of points in the user coordinate system, a group identifier, a left area identifier, and a right area identifier.

All routines in the package AREAS have names that begin with "AR."

**Step One**

To initialize AREAS, execute the FORTRAN statement

```
CALL ARINAM (IAM, LAM)
```

LAM is unchanged by the call.

IAM An integer array in which an area map is to be constructed.

LAM Length of the IAM array.

("ARINAM" stands for initialize area map.)

**Step Two**

For each edge, execute the FORTRAN statement

```
CALL AREDAM (IAM, XCA, YCA, LCA, IGI, IDL, IDR)
```

All arguments except IAM are input arguments and are unchanged by the call.

IAM The area-map array.

XCA Real array holding the X coordinates, in the user coordinate system, of the points defining the edge.

YCA Real array holding the Y coordinates, in the user coordinate system, of the points defining the edge.

LCA Number of coordinates in XCA and YCA.
IGI Identifier of the group to which the edge belongs.
IDL The identifier of the area that lies to the left of the edge.
IDR Identifier of the area that lies to the right of the edge.

("AREDAM" stands for edge to area map.)

Step Three Once all of the edges have been inserted in the area map, the area map must be preprocessed. You can do this by executing the FORTRAN statement

```
CALL ARPRAM (IAM, IF1, IF2, IF3)
```

Note: If you do not call ARPRAM, any of the routines ARSCAM, ARDRLN, or ARGTAI will call it. However, the routines will set the shortcut flags IF1, IF2, and IF3 to not take any shortcuts, which may result in the subroutine running longer than is necessary. For the details of using the flags, see ARPRAM in the "Details of Routines" section below.

All arguments except IAM are input arguments and are unchanged by the call.

IAM The area-map array.
IF1, IF2, IF3 Flags that designate what shortcuts are allowed in the preprocessing step.

("ARPRAM" stands for preprocess area map.)

USES OF AN AREA MAP Once you have completed the preceding three steps, you can use the resulting area map for three different purposes:

- to obtain the definitions of areas.
- to mask objects being drawn.
- to get the area identifiers associated with a given point.

Each of these uses is explained briefly below. For more information, see the "Details of Routines" section later in this document.

Obtaining Area Definitions To obtain the definitions of the areas created by the edges inserted in the area map, (to color the area, for example) execute the following FORTRAN statement:

```
CALL ARSCAM (IAM, XCS, YCS, MCS, IAI, IAG, MAI, APR)
```

All arguments except IAM are input arguments and are unchanged by the call.

IAM The area-map array.
XCS Real array of size MCS.
YCS Real array of size MCS.
MCS  Dimension of each of the arrays XCS and YCS.
IAI  Integer array of size MAI.
IAG  Integer array of size MAI.
MAI  Dimension of each of the arrays IAI and IAG.

APR  An area-processing routine you supply. The name of APR must
appear in an EXTERNAL statement in the routine that calls
ARSCAM. ARSCAM scans the area map from left to right,
picking off, one by one, the areas defined by it. ARSCAM calls
the routine APR to process each defined area. For details of the
structure of the routine APR, see ARSCAM in the "Details of
Routines" section below.

("ARSCAM" stands for scan area map.)

Masking  The routine ARDRLN is used to draw a polyline that is masked by a
given area map. (A polyline is a line defined by two or more points.)
Given a polyline and an area map, ARDRLN breaks the polyline into
pieces in such a way that each piece lies within only one area. For each
such piece of the polyline, a routine specified by you is called. That rou-
tine may draw the piece using a specified color, intensity, and dash pat-
tern (or the it may not draw the line at all). For example, ARDRLN can
be used to limit the drawing of latitude and longitude lines to areas over
the ocean, omitting the lines over land. In a future contouring package,
ARDRLN will be used to keep contour lines from passing through contour
labels.

ARDRLN is called using a FORTRAN statement such as

    CALL ARDRLN (IAM, XCD, YCD, NCD, XCS, YCS, MCS, IAI, IAG, MAI, LPR)

IAM  The area-map array.
XCD  Real array holding the X coordinates, in the user coordinate sys-
tem, of the points defining the polyline.
YCD  Real array holding the Y coordinates, in the user coordinate sys-
tem, of the points defining the polyline.
NCD  Number of points defining the polyline.
XCS  Real array of size MCS; used in calls to LPR.
YCS  Real array of size MCS; used in calls to LPR.
MCS  Dimension of each of the arrays XCS and YCS.
IAI  Integer array of size MAI; used in calls to LPR.
IAG  Integer array of size MAI; used in calls to LPR.
MAI  Dimension of each of the arrays IAI and IAG.
AREAS

LPR. Line-processing routine that is supplied by you, which must be declared EXTERNAL in the routine calling ARDRLN. For details of the structure of the routine LPR, see ARDRLN in the "Details of Routines" section below.

("ARDRLN" stands for draw line.)

Obtaining Area Identifiers

The routine ARGTAI is used to get the area identifiers associated with a given point. ARGTAI is called using a FORTRAN statement such as

CALL ARGTAI (IAM, XCD, YCD, IAI, IAG, MAI, NAI, ICF)

IAM The area-map array.
XCD X coordinate, in the current user coordinate system, of a point about which you want to obtain information.
YCD Y coordinate, in the current user coordinate system, of a point about which you want to obtain information.
IAI Integer array of size MAI, to which information about the specified point will be returned.
IAG Integer array of size MAI, to which information about the specified point will be returned.
MAI Dimension of each of the arrays IAI and IAG.
NAI Number of values returned in IAI and IAG. NAI will be equal to the number of groups of edges that you put in the area map.
ICF Flag set non-zero by you to indicate that the definition of the user coordinate system has been changed since the last call to ARGTAI, in which case calls to GETSET must be executed by ARGTAI. If you set the flag to zero, it will be assumed that the information retrieved previously is still correct and that calls to GETSET may be skipped.

("ARGTAI" stands for get area identifiers.)

DETAILS OF ROUTINES

This section contains detailed descriptions of all the routines in AREAS. The routines are arranged in alphabetical order.

ROUTINE ARDRLN

Latest Revision June 1987
Purpose
ARDRLN is given an existing area map and a polyline. ARDRLN breaks the polyline into pieces, each of which lies entirely within a single area defined by the area map. It calls a user routine LPR once for each piece. You can program LPR to do whatever you wish with the pieces. For example, you can change the color, intensity, or dash pattern before drawing a given piece — or you may choose not to draw a given piece at all.

Usage
CALL ARDRLN (IAM, XCD, YCD, NCD, XCS, YCS, MCS, IAI, IAG, MAI, LPR)

Type and Dimension of Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM</td>
<td>INTEGER</td>
<td>IAM(*)</td>
</tr>
<tr>
<td>XCD, YCD</td>
<td>REAL</td>
<td>XCS(MCS), YCS(MCS)</td>
</tr>
<tr>
<td>NCD</td>
<td>INTEGER</td>
<td>NCD</td>
</tr>
<tr>
<td>XCS, YCS</td>
<td>REAL</td>
<td>MCS</td>
</tr>
<tr>
<td>MCS</td>
<td>INTEGER</td>
<td>MAI</td>
</tr>
<tr>
<td>IAI, IAG</td>
<td>INTEGER</td>
<td>IAI(MAI), IAG(MAI)</td>
</tr>
<tr>
<td>LPR</td>
<td>EXTERNAL</td>
<td>LPR</td>
</tr>
</tbody>
</table>

Arguments
IAM
An integer array containing an area map. IAM must have been initialized by a call to ARINAM and edges must have been put into it by calls to AREDAM. If you did not preprocess the area map by calling ARPRAM, ARDRLN will call ARPRAM before doing anything else.

XCD, YCD
Real arrays. The first NCD elements are the X and Y coordinates, in the user coordinate system, of points defining a polyline in the user coordinate system.

NCD
Number of coordinates in XCD and YCD.

XCS, YCS
Real arrays, each dimensioned MCS, for use by ARDRLN in calls to the line processing routine LPR.

MCS
Dimension of each of the arrays XCS and YCS.

IAI, IAG
Integer arrays, each dimensioned MAI, for use by ARDRLN in calls to the line processing routine LPR.

MAI
Dimension of each of the arrays IAI and IAG.

LPR
A line-processing routine, declared EXTERNAL in the routine which calls ARDRLN. You must supply the routine LPR and it must have the following form:

```fortran
SUBROUTINE LPR (XCS, YCS, NCS, IAI, IAG, MAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)

(CODE TO PROCESS POLYLINE DEFINED BY XCS, YCS, IAI, AND IAG)

RETURN
END
```

XCS, YCS
Hold the X and Y coordinates, in the fractional coordinate system, of NCS points defining a piece of the original polyline.

NCS
Number of X and Y coordinates in the arrays XCS and YCS.
AREAS

IAI, IAG Hold NAI pairs of identifiers for the area within which the piece of the polyline lies. For each value of I from 1 to NAI, IAI(I) is the area identifier for the area with respect to the group of edges specified by the group identifier IAG(I).

NAI Number of values in IAI and IAG. NAI will be equal to the number of groups of edges that you put in the area map.

Before executing the first call to LPR, ARDRLN calls GETSET to retrieve the current user-system mapping parameters and then executes the statement

``CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)``

This ensures that, if the normalized device coordinates in XCS and YCS are used in calls to such routines as GPL and CURVE, the results will be correct. LPR may do its own SET call to achieve some other effect. Before returning control to the calling routine, ARDRLN calls SET again to restore the original mapping parameters.

**ROUTINE AREDAM**

Latest Revision June 1987

Purpose AREDAM is called once for each edge that is to be added to a given area map.

Usage CALL AREDAM (IAM, XCA, YCA, LCA, IGI, IDL, IDR)

If an edge is too long to be conveniently handled by a single call to AREDAM, the edge may be broken into two or more pieces, but the pieces must fit together to define the whole edge. Each piece following the first must start with the last point of the preceding edge. For example, to approximate a circle with 360 points, you could do a single call defining a 361-point edge in which the last point was a duplicate of the first, or you could do two calls, the first using points 1 through 181, and the second using points 181 through 361, of the original edge.

**Type and Dimension of Arguments**

- INTEGER IAM(*)
- INTEGER LCA
- INTEGER IDL
- REAL XCA(*)
- INTEGER IGI
- INTEGER IDR
- REAL YCA(*)

Arguments

IAM The integer array in which the area map is being constructed. IAM must have been initialized previously by a call to the routine ARINAM.

XCA, YCA Real arrays. The first LCA elements are the X and Y coordinates of the points that define the edge to be added
to the area map in the current user coordinate system. Note that not all edges have to be defined in the same user coordinate system (generally, though, all the edges in a particular group will be).

LCA Specifies the number of points defining the edge. The order in which the points are specified determines the meaning of "left" and "right." "Left" and "right" are defined from the standpoint of an observer at the position of point 1, looking in the direction of point 2.

IGI Group identifier for the group of edges to which the edge belongs.

IDL Area identifier for the area to the left of the edge.

IDR Area identifier for the area to the right of the edge.

ROUTINE ARGTAI

Latest Revision June 1987

Purpose Given an area map and the user-system coordinates of a point, the routine ARGTAI returns information about the area containing that point.

Usage CALL ARGTAI (IAM,XCD,YCD, IAI, IAG, MAI, NAI, ICF)

Type and Dimension of Arguments

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Type</th>
<th>Dimension</th>
</tr>
</thead>
</table>

IAM INTEGER IAM(*) INTEGER IAI(MAI) INTEGER NAI
REAL XCD INTEGER IAG(MAI) INTEGER ICF
REAL YCD INTEGER MAI

Arguments IAM An integer array containing an area map that has been initialized by a call to ARINAM and to which edges have been added by calls to AREXAM. If you did not preprocess the area map by calling ARPRAM, ARGTAI will call ARPRAM before doing anything else.

XCD, YCD The X and Y coordinates, in the current user coordinate system, of a point about which information is desired. Note that the current user coordinate system need not be the one that was current while the area map was being constructed.

IAI, IAG Integer arrays, dimensioned MAI, in which information will be returned by ARGTAI. For each value of I from 1 to NAI, IAI(I) is the area identifier and IAG(I) is the associated group identifier for the area containing the point (XCD,YCD). Note that the entries in IAI and IAG may not be assumed to be in any particular order; if group identifiers 1, 2, and 3 were used, one of the set (IAG(1),IAG(2),IAG(3)) will be a 1, one will be a 2, and one will be a 3, but they may be in any of six different orders.

MAI Dimension of the arrays IAI and IAG.
 AREAS

NAI Set by ARGTAI to indicate how many entries have been placed in the arrays IAI and IAG. NAI will be equal to the number of groups of edges that you put in the area map.

ICF Flag set non-zero by you to indicate that the definition of the user coordinate system has been changed since the last call to ARGTAI, in which case calls to GETSET must be executed by ARGTAI. If you set the flag to zero, ARGTAI assumes that the information retrieved previously is still correct and that calls to GETSET may be skipped. This becomes important when you make many calls to ARGTAI and you want to avoid the overhead associated with the calls to GETSET.

ROUTINE ARINAM

Latest Revision June 1987
Purpose The routine ARINAM is called to initialize an area map.
Usage CALL ARINAM (IAM, LAM)
Type and Dimension of Arguments
INTEGER IAM(LAM)
INTEGER LAM
Arguments
IAM An integer array in which an area map is to be constructed. Each original point requires ten words in the array IAM and additional points will be added to these; therefore, the array needs to be large.
LAM Length of the array IAM.

ROUTINE ARPRAM

Latest Revision June 1987
Purpose ARPRAM is called to preprocess an area map that has been initialized by a call to ARINAM and to which edges have been added by calls to AREDAM.
Usage CALL ARPRAM (IAM, IF1, IF2, IF3)
If you call ARPRAM, you must do it before you call any of the routines ARDRLN, ARGTAI, or ARSCAM. If you did not call ARPRAM, each of these routines will call it before doing anything else. The only advantage in calling ARPRAM yourself is that you may be able to set some of the arguments to make it work more efficiently. In order to understand how to set the arguments IF1, IF2, and IF3, you need to understand what ARPRAM does:
1. ARPRAM first shortens edge segments whose projections on the X axis are more than twice as long as the average. ARPRAM does this by interpolating points along their lengths. This leads to greater efficiency in executing other parts of the algorithm.

2. Next, ARPRAM finds all intersections of edges with each other and interpolates points along the edge segments that intersect; this can be a time-consuming operation.

   IF1 If you set IF1 non-zero, ARPRAM examines a pair of edge segments only if one of the pair has a left or right area identifier that is zero or negative. This would be appropriate, for example, for contour lines, which are known not to intersect each other, but only to intersect the perimeter.

3. Next, ARPRAM searches for and removes "dangling" edges (those that do not contribute to enclosing any area); such edges are removed.

   IF2 If you set IF2 non-zero, ARPRAM skips this process. Again, this would be appropriate for contour lines, which are known not to have any such edges. This would not be appropriate for EZMAP boundary lines. (The EZMAP dataset contains, for example, islands that are formed from simple, unclosed curves.)

4. Next, ARPRAM adjusts area-identifier information in the area map. It examines all the edge segments of each area in each group to see what area identifier should be assigned to the area, and makes the following types of adjustments:

   - If a negative value is found anywhere, the area is considered to have area identifier -1.
   - If only zeroes are found, the area is considered to have area identifier 0.
   - If no negatives are found and at least one value greater than zero is found, the greater-than-zero value belonging to the edge most recently inserted in the area map is used.

   In any case, all of the area identifiers for a given area are reset to the same value, so that all the information will be consistent.

   IF3 If you set IF3 non-zero, ARPRAM speeds up this process by omitting the consideration of "holes" in the areas examined. Again, this is appropriate for contour lines.

5. In every case, if an internal problem is detected that seems to indicate that a shortcut was taken inappropriately, ARPRAM attempts to fix the problem by re-executing the code with the shortcut turned off.
Type and Dimension of Arguments

Arguments IAM

INTEGER IAM(*) INTEGER IF2 INTEGER IF3
INTEGER IF1

Arguments IF1, IF2, IF3

An integer array containing an area map that has been initialized by a call to ARINAM and to which edges have been added by calls to AREDAM.

Shortcut flags, as described above. In every case, the value 0 indicates that ARPRAM will not take a particular shortcut; any other value indicates that ARPRAM will take the shortcut.

ROUTINE ARSCAM

Latest Revision June 1987

Purpose Given an existing area map, the routine ARSCAM scans it from left to right, extracting the definitions of all the areas formed by the area map.

Usage CALL ARSCAM (IAM, XCS, YCS, MCS, IAI, IAG, MAI, APR)

ARSCAM calls a user routine APR once for each scanned area; APR may be programmed to do whatever you want with each area (color it, fill it with cross-hatch marks, or save it in a file, for example).

Type and Dimension of Arguments

Arguments IAM

INTEGER IAM(*) INTEGER MCS INTEGER MAI
REAL XCS(MCS) INTEGER IAI(MAI) EXTERNAL APR
REAL YCS(MCS) INTEGER IAG(MAI)

Arguments XCS, YCS

Real arrays, each dimensioned MCS, for use by ARSCAM in calls to the area processing routine APR.

MCS Dimension of each of the arrays XCS and YCS.

IAI, IAG Integer arrays, each dimensioned MAI, for use by ARSCAM in calls to the area processing routine APR.

MAI Dimension of each of the arrays IAI and IAG.

APR An area processing routine, declared EXTERNAL in the routine that calls ARSCAM. You must supply the routine APR and it must have the following form:

SUBROUTINE APR (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)

(CODE TO PROCESS THE AREA DEFINED BY XCS, YCS, IAI, AND IAG)

August 1987
Hold the X and Y coordinates, in the fractional coordinate system, of NCS points defining a polygonal area. The last of these points is a duplicate of the first. Holes in an area are traced in such a way as to maximize the probability of hardware fill working properly by using vertical lines to get to and from the holes and tracing them in the proper direction.

Number of X and Y coordinates in the arrays XCS and YCS.

Hold NAI pairs of identifiers for the area defined by XCS and YCS. For each value of I from 1 to NAI, IAI(I) is the area identifier for the area with respect to the group of edges specified by the group identifier IAG(I).

Number of values in IAI and IAG. NAI will be equal to the number of groups that you put in the area map.

Before executing the first call to APR, ARSCAM calls GETSET to retrieve the current user-system mapping parameters and then executes the statement

\[
\text{CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)}
\]

This ensures that, if the normalized device coordinates in XCS and YCS are used in calls to such routines as GFA and FILL, the results will be correct. APR may do its own SET call to achieve some other effect. Before returning control to the calling routine, ARSCAM re-calls SET to restore the original mapping parameters.

**Error Messages**

When the package detects errors, it calls the routine SETER. As of this writing, all errors are treated as recoverable. The following error messages will be issued to the standard diagnostic unit. You will probably never see the diagnostic "ALGORITHM FAILURE," but, with a new package, such problems are possible. If you get the "ALGORITHM FAILURE" message, contact your NCAR Graphics site representative.

\[
\text{ARDRLN - AREA-MAP ARRAY UNINITIALIZED}
\text{ARDRLN - MAI TOO SMALL}
\text{ARDRLN - ALGORITHM FAILURE}
\text{AREDAM - AREA-MAP ARRAY UNINITIALIZED}
\text{AREDAM - AREA-MAP ARRAY OVERFLOW}
\]
Special Problems

On some devices, there is an upper limit on the number of points that can be used to define a polygon to be color-filled. If the areas in which you are interested are intrinsically more complicated than the limit allows, you may overlay the areas with a group of edges defining a set of vertical strips. (Because of the way the code works, you should not use horizontal strips, which will result in seriously reduced efficiency.) Since the area identifier is not important, choose any non-negative value for use as the area identifiers for all of the strips. Using vertical strips reduces the number of points defining each area. However, it also increases the run time of AREAS and the time required to draw any resulting plot.

Using AREAS with EZMAPA

The mapping utility, EZMAP, has been extended to use AREAS to create solid-colored maps. See the documentation for EZMAPA for instructions.
AUTOGRAPH
AUTOGRAPH

Following is a brief description of the AUTOGRAPH package. For complete documentation, see the manual, AUTOGRAPH: A Graphing Utility, which was shipped to you along with this manual.

Important Note
If you have existing programs that use the FORTRAN 66 version of AUTOGRAPH (any version of AUTOGRAPH acquired before October 23, 1984), you must first convert them to use the FORTRAN 77 version of AUTOGRAPH. In the manual, AUTOGRAPH: A Graphing Utility, see Appendix A, "Conversion from FORTRAN 66 AUTOGRAPH to FORTRAN 77 AUTOGRAPH," for instructions.

Latest Revision
August 1987

Purpose
To draw graphs, each with a labeled background and each displaying one or more curves.

Access (on the CRAY)
To use AUTOGRAPH routines on the CRAY, simply call them; they are in the binary library $NCARLB, which is automatically searched.

To get smoother curves, drawn using spline interpolation, compile DASHSMTH, from ULIB, to replace DASHCHAR, from $NCARLB:

GETSRC,LIB=ULIB,FILE=DASHSMTH,L=DSMTH.
CFT,I=DSMTH,L=0.

AUTOGRAPH contains a routine AGPWRT, which it calls to draw labels. This routine just passes its arguments on to the system-plot-package routine PWRIT. To use one of the fancier character-drawers, like PWRITX or PWRITY, just compile a routine AGPWRT to replace the default version; it has the same arguments as PWRIT and may either draw the character string itself, or just pass the arguments on to a desired character-drawer. AUTOGRAPH contains a subroutine AGPWRT that you can modify so that AUTOGRAPH will access character fonts that are different from the default font. In its distributed form, AGPWRT is a dummy subroutine. Consult the documentation in the code. The file AGUPWRTX on the distribution tape contains an implementation of AGPWRTX that allows AUTOGRAPH to access the PWRITX character set. See the documentation in the code for AGUPWRT. See your NCAR Graphics site representative to learn how to access the code.

Usage
Following this indented preamble are given two lists: one describing the AUTOGRAPH routines and another describing the arguments of those routines.

AUTOGRAPH: A Graphing Utility gives a complete write-up of AUTOGRAPH, in great detail and with a set of helpful examples.
Entry Points

Except for seven routines, which are included in the package for historical reasons (EZY, EZXY, EZMY, EZMXY, IDIOT, ANOTAT, and DISPLA), the AUTOGRAPH routines have six-character names beginning with the characters AG. An alphabetized list follows:

- AGAXIS
- AGBACK
- AGBNCH
- AGCHAX
- AGCHCU
- AGCHIL
- AGCTCS
- AGCURV
- AGDASH
- AGDFLT
- AGDLCH
- AGDSHN
- AGEXAX
- AGEXUS
- AGFPBN
- AGFTOL
- AGGETC
- AGGETF
- AGGETI
- AGGETP
- AGGTCH
- AGINIT
- AGKURV
- AGMAXI
- AGMINI
- AGNUMB
- AGPPID
- AGQURV
- AGPNBN
- AGQURV
- AGRPCH
- AGSTCH
- AGSAVE
- AGSCAN
- AGSRCH
- AGSTUP
- AGSTUP
- AGSTUP
- AGSTUP
- AGSTUP
- AGSTUP
- AGSTUP
- AGSTUP
- AGUTOL

NOTE: The routine AGDFLT is a block-data routine specifying the default values of AUTOGRAPH control parameters.

Special Conditions

Under certain conditions, AUTOGRAPH may print an error message (via the routine SETER) and stop. Each error message includes the name of the routine which issued it. A description of the condition which caused the error may be found in the AUTOGRAPH write-up in the NCAR graphics manual; look in the write-up of the routine which issued the error message, under the heading 'SPECIAL CONDITIONS.'

For error messages issued by the routine AGNUMB, see the write-up of the routine AGSTUP.

If you get an error in the routine ALOG10, it probably means that you are using a logarithmic axis and some of the coordinate data along that axis are zero or negative.

Common Blocks

The AUTOGRAPH common blocks are AGCONP, AGORIP, AGOCHP, AGCHR1, and AGCHR2. AGCONP contains the AUTOGRAPH "control parameters", primary and secondary, all of which are real, AGORIP other real and/or integer parameters, AGOCHP other character parameters, AGCHR1 and AGCHR2 the variables implementing the character-storage-and-retrieval scheme of AUTOGRAPH.

I/O

Lower-level plotting routines are called to produce graphical output and, when errors occur, error messages may be written to the system error file, as defined by I1MACH(4), either directly or by way of a call to SETER.

Required ULIB Routines

AUTOGRAPH uses the software dashed-line package DASHCHAR. Of course, either of the packages DASHSMTH or DASHSUPR may be used instead, to get smoother curves.

Language

FORTRAN 77
History

Dave Robertson wrote the original routine IDIOT, which was intended to provide a simple, quick-and-dirty, x-y graph-drawing capability. In time, as it became obvious that many users were adapting IDIOT to more sophisticated tasks, Dan Anderson wrote the first AUTOGRAPH package, based on IDIOT. It allowed the user to put more than one curve on a graph, to use more sophisticated backgrounds, to specify coordinate data in a variety of ways, and to more easily control the scaling and positioning of graphs. Eventually, this package, too, was found wanting. In 1977, Dave Kennison entirely re-wrote AUTOGRAPH, with the following goals: to maintain the ease of use for simple graphs which had been the principal virtue of the package, to provide the user with as much control as possible, to incorporate desirable new features, and to make the package as portable as possible. In 1984, the package was again worked over by Dave Kennison, to make it compatible with FORTRAN 77 and to remove any dependency on the LOC function, which had proved to cause difficulties on certain machines. The user interface was changed somewhat and some new features were added. A GKS-compatible version was written.

Space Required

AUTOGRAPH is big; one pays a price for its capabilities. On the CRAY, it occupies a little under 30000 (octal) locations. The required plot package routines take about another 7000 (octal), the (modified) PORT support routines about another 1000 (octal), and system routines (math, I/O, miscellany) another 30000 (octal).

Portability

AUTOGRAPH may be ported with few modifications to most systems having a FORTRAN 77 compiler.

The labeled common blocks may have to be declared in a part of the user program which is always core-resident so that variables in them will maintain their values from one AUTOGRAPH-routine call to the next. Such a problem may arise when AUTOGRAPH is placed in an overlay or when some sort of memory-paging scheme is used.

Required Resident Routines

AUTOGRAPH uses the DASHCHAR routines DASHDB, DASHDC, FRSTD, LASTD, LINED, AND VECTD, the system-plot-package routines FRAME, GETSET, GETSI, LINE, PWRIT, and SET, the support routines ISHIFT and IOR, the (modified) PORT utilities SETER and I1MACH, and the FORTRAN-library routines ALOG10, ATAN2, COS, SIN, AND SQRT.

Required GKS Level

0A

User-Callable AUTOGRAPH Routines

Following is a list of AUTOGRAPH routines to be called by the user (organized by function). Each routine is described briefly. The arguments of the routines are described in the next section.

Each of the following routines draws a complete graph with one call. Each is implemented by a set of calls to the lower-level AUTOGRAPH routines AGSTUP, AGCURV, and AGBACK (see, below).
- **EZY (YDRA,NPTS,GLAB)** — draws a graph of the curve defined by the data points
  
  \[ ((I,YDRA(I)), I=1,NPTS), \]

  with a graph label specified by GLAB.

- **EZXY (XDRA,YDRA,NPTS,GLAB)** — draws a graph of the curve defined by the data points
  
  \[ ((XDRA(I),YDRA(I)), I=1,NPTS), \]

  with a graph label specified by GLAB.

- **EZMY (YDRA,IDX,Y,MANY,NPTS,GLAB)** — draws a graph of the family of curves defined by data points
  
  \[ (((I,YDRA(I,J)), I=1,NPTS), J=1,MANY), \]

  with a graph label specified by GLAB. The order of the subscripts of YDRA may be reversed — see the routine DISPLA, argument LROW.

- **EZMXY (XDRA,YDRA,IDX,Y,MANY,NPTS,GLAB)** — draws a graph of the family of curves defined by the data points
  
  \[ (((XDRA(I),YDRA(I,J)), I=1,NPTS), J=1,MANY), \]

  with a graph label specified by GLAB. XDRA may be doubly-subscripted and the order of the subscripts of XDRA and YDRA may be reversed — see the routine DISPLA, argument LROW.

- **IDIOT (XDRA,YDRA,NPTS,LTYP,LDSH,LABX,LABY,LABG,LFRA)** — implements the routine from which AUTOGRAPH grew — not recommended — provided for antique lovers.

The following routines provide user access to the AUTOGRAPH control parameters (in the labeled common block AGCONP).

- **ANOTAT (XLAB,YLAB,LABC,LSET,NDSH,DSDL)** — may be used to change the x- and y-axis (non-numeric) labels, the background type, the way in which graphs are positioned and scaled, and the type of dash patterns to be used in drawing curves.

- **DISPLA (LFRA,LROW,LTYP)** — may be used to specify when, if ever, the EZ... routines do a frame advance, how input arrays for EZMY and EZMXY are dimensioned, and the linear/log nature of graphs.

- **AGSETP (TPGN,FURA,LURA)** — a general-purpose parameter-setting routine, used to set the group of parameters specified by TPGN, using values obtained from the array
  
  \[ (FURA(I), I=1,LURA). \]

- **AGSETF (TPGN,FUSR)** — used to set the single parameter specified by TPGN, giving it the floating-point value FUSR.

- **AGSETI (TPGN,IUSR)** — used to set the single parameter specified by TPGN, giving it the floating-point value FLOAT(IUSR).

- **AGSETC (TPGN,CUSR)** — the character string CUSR is stashed in an array inside AUTOGRAPH and the floating-point equivalent of an identifier which may be used for later retrieval of the string is stored as the value of the single parameter specified by TPGN.
The single parameter must be a label name, a dash pattern, the
text of a label line, or the line-terminator character.

- **AGGETP (TPGN,FURA,LURA)** — a general-purpose parameter-getting routine, used to get the group of parameters specified by TPGN, putting the result in the array (FURA[],I=1:LURA).

- **AGGETF (TPGN,FUSR)** — used to get, in FUSR, the floating-point value of the single parameter specified by TPGN.

- **AGGETI (TPGN,IUSR)** — used to get, in IUSR, the integer equivalent of the value of the single parameter specified by TPGN.

- **AGGETC (TPGN,CUSR)** — used to get, in CUSR, the character string whose identifier is specified by the integer equivalent of the single parameter specified by TPGN. The single parameter must be a label name, a dash pattern, the text of a label line, or the line-terminator character.

The following are lower-level routines, which may be used to draw graphs of many different kinds. The EZ... routines call these. They are intended to be called by user programs, as well.

- **AGSTUP (XDXRA,NVIX,IVIX,NEVX,IIEX,YDRA,NVIX,IVYX,NEVY,IIEY)** — this routine must be called prior to the first call to either of the two routines AGBACK and AGCURV, to force the set-up of secondary parameters controlling the behavior of those routines. After any parameter-setting call, AGSTUP must be called again before calling either AGBACK or AGCURV again. AGSTUP calls the routine "SET", in the plot package, so that user x/y coordinates in subsequent calls will map properly into the plotter space.

- **AGBACK** — draws the background defined by the current state of the AUTOGRAPH control parameters.

- **AGCURV (XVEC,IIEX,YVEC,IIEY,NEXY,KDSH)** — draws the curve defined by the arguments, positioning it as specified by the current state of the AUTOGRAPH control parameters.

The following utility routines are called by the user.

- **AGSAVE (IFNO)** — used to save the current state of AUTOGRAPH by writing the appropriate information to a specified file. Most commonly used to save the default state for later restoration. This routine should be used instead of AGGETP when the object is to save the whole state of AUTOGRAPH, since it saves not only the primary control parameters, but all of the character strings pointed to by the primary control parameters. It is the user's responsibility to position the file before calling AGSAVE.

- **AGRSTR (IFNO)** — used to restore a saved state of AUTOGRAPH by reading the appropriate information from a specified file. Most commonly used to restore AUTOGRAPH to its
default state. It is the user's responsibility to position the file before calling AGRSTR.

- **AGBNCH (IDSH)** — a function, of type CHARACTER*16 (it must be declared as such in a user routine referencing it), whose value, given a 16-bit binary dash pattern, is the equivalent character dash pattern.

- **AGDSHN (IDSH)** — a function, of type CHARACTER*16 (it must be declared as such in a user routine referencing it), whose value, given an integer "n" (typically between 1 and 26) is the character string 'DASH/ARRAY/nnnn.', which is the name of the nth dash pattern parameter. To set the 13th dash pattern, for example, one might use "CALL AGSETC (AGDSHN(13),$$$$$$CURVE 13$$$$$$)".

The following utility routines are called by AUTOGRAPH. The versions included in AUTOGRAPH itself are dummies; they do nothing but RETURN. The user may replace one or more of these routines with versions to accomplish specific purposes.

- **AGUTOL (IAXS,FUNS,IDMA,VINP,VOTP)** — called by AUTOGRAPH to perform the mapping from user-system values along an axis to label-system values along the axis and vice-versa. This routine may be replaced by the user to create a desired graph.

- **AGCHAX (IFLG,IAXS,IPRT,VILS)** — called by AUTOGRAPH just before and just after the various parts of the axes are drawn.

- **AGCHCU (IFLG,KDSH)** — called by AUTOGRAPH just before and just after each curve is drawn.

- **AGCHIL (IFLG,LBNM,LNNO)** — called by AUTOGRAPH just before and just after each line of an informational label is drawn.

- **AGCHNL (IAXS,VILS,CHRM,MCIM,NCIM,IPXM,CHRE,MCIE,NCIE)** — called by AUTOGRAPH just after the character strings defining a numeric label have been generated.

**Arguments**

In calls to the routines EZY, EZXY, EZMY, and EZMXY:

- **XDRA** is an array of x coordinates, dimensioned as implied by the current value of the AUTOGRAPH control parameter 'ROW.' (see the description of the argument LROW, below). The value of the AUTOGRAPH parameter 'NULL/I.' (1.E36, by default) when used as an x coordinate, implies a missing data point; the curve segments on either side of such a point are not drawn.

- **YDRA** is an array of y coordinates, dimensioned as implied by the current value of the AUTOGRAPH control parameter 'ROW.' (see the description of the argument LROW, below). The value of the AUTOGRAPH parameter 'NULL/I.' (1.E36, by default) when used as a y coordinate, implies a missing data point; the curve segments on either side of such a point are not drawn.
• IDXY is the first dimension of the arrays XDRA (if it has two dimensions) and YDRA.

• MANY is the number of curves to be drawn by the call to EZ . . . — normally, the second dimension of XDRA (if it has two dimensions) and YDRA.

• NPTS is the number of points defining each curve to be drawn by the routine EZ . . . — normally, the first (or only) dimension of XDRA and YDRA.

• GLAB is a character constant or a character variable, defining a label to be placed at the top of the graph. The string may not be more than 40 characters long — if it is fewer than 40 characters long, its last character must be a dollar sign. (The dollar sign is not a part of the label — it is stripped off.) The character string "CHAR(0)" may be used to indicate that the previous label, whatever it was, should continue to be used. The initial graph label consists of blanks.

In calls to the routine ANOTAT:

• XLAB and YLAB resemble GLAB (see above) and define labels for the x and y axes. The default x-axis label is the single character X, the default y-axis label the single character Y. Note that one may use the string "CHAR(0)" to indicate that the x-axis (y-axis) label is not to be changed from what it was previously.

• LBAC, if non-zero, specifies a new value for the AUTOGRAPH control parameter 'BACKGROUND.', as follows:

1 a perimeter background
2 a grid background
3 an axis background
4 no background

The default value of 'BACKGROUND.' is 1.

• LSET, if non-zero, specifies a new value for the AUTOGRAPH control parameter 'SET.'. This parameter may be negated to suspend the drawing of curves by the EZ . . . routines, so that a call to one of them will produce only a background. The absolute value of 'SET.' affects the way in which AUTOGRAPH determines the position and shape of the graph and the scaling of the axes, as follows:

1 Restores the default values of the AUTOGRAPH parameters in question. AUTOGRAPH will set up an appropriate call to the plot-package routine "SET", over-riding any prior call to that routine.

2 Tells AUTOGRAPH to use arguments 1-4 and 9 of the last "SET" call. Arguments 1-4 specify where the graph should fall on the plotter frame, argument 9 whether the graph is linear/linear, linear/log, etc.
3 Tells AUTOGRAPH to use arguments 5-8 and 9 of the last "SET" call. Arguments 5-8 specify the scaling of the axes, argument 9 whether the graph is linear/linear, linear/log, etc.

4 A combination of 2 and 3. Arguments 1-4 of the last "SET" call specify the position, arguments 5-8 the scaling, and argument 9 the linear/log nature, of the graph.

(The plot-package routine "SET" is described in the "SPPS" section of this manual; it is not a part of AUTOGRAPH.)

If the routine DISPLA is called with its argument LTYP non-zero, the linear/log nature of the graph will be that specified by LTYP, not that specified by the last "SET" call, no matter what the value of the control parameter 'SET'.

The default value of 'SET.' is 1.

- NDSH, if non-zero, specifies a new value of the AUTOGRAPH control parameter 'DASH/SELECTOR.' (and therefore a new set of dashed-line patterns), as described below. Note: The default value of the dashed-line parameters is such that all curves will be drawn using solid lines; if that is what you want, use a zero for NDSH.

  If the value of 'DASH/SELECTOR.' is negative, curves produced by subsequent calls to EZMY or EZMXY will be drawn using a set of alphabetic dashed-line patterns. The first curve drawn by a given call will be labeled 'A', the second 'B', . . ., the twenty-sixth 'Z', the twenty-seventh 'A' again, and so on. Curves drawn by calls to EZY and EZXY will be unaffected.

  If the value of 'DASH/SELECTOR.' is positive, it must be less than or equal to 26. The next argument, DSHL, is an array containing NDSH dashed-line patterns. All curves produced by subsequent calls to EZY, EZXY, EZMY, and EZMXY will be drawn using the dashed-line patterns in (DSHL(I),I=1,NDSH) — the first curve produced by a given call will have the pattern specified by DSHL(1), the second that specified by DSHL(2), the third that specified by DSHL(3), . . . the NDSH+1st that specified by DSHL(1), . . . etc. Each element of DSHL must be a character string, in which a dollar sign stands for a solid-line segment, a quote stands for a gap, and other characters stand for themselves. See the write-up of the package "DASHCHAR" in this manual. Binary dashed-line patterns may not be defined by means of a call to ANOTAT, only by means of calls to lower-level routines.

- DSHL (if NDSH is greater than zero) is an array of dashed-line patterns, as described above.

In calls to the routine DISPLA:

- LFRA, if non-zero, specifies a new value for the AUTOGRAPH control parameter 'FRAME.' Possible values are as follows:
1 The EZ... routines do a frame advance after drawing.
2 No frame advance is done by the EZ... routines.
3 The EZ... routines do a frame advance before drawing.

The default value of 'FRAME.' is 1.

- **LROW**, if non-zero, specifies a new value for the AUTOGRAPH control parameter 'ROW.'. This parameter tells AUTOGRAPH how the argument arrays XDRA and YDRA, in calls to the routines EZMY and EZMXY, are subscripted, as follows:

  If 'ROW.' is positive, this implies that the first subscript of YDRA is a point number and the second subscript is a curve number. If 'ROW.' is negative, the order is reversed.

  If the absolute value of 'ROW.' is 1, this implies that XDRA is singly-subscripted, by point number only. If the absolute value of 'ROW.' is 2 or greater, this implies that XDRA is doubly-subscripted, just like YDRA.

  The default value of 'ROW.' is 1, specifying that XDRA is singly-subscripted and that YDRA is doubly-subscripted by point number and curve number, in that order.

- **LTYP**, if non-zero, specifies new values for the AUTOGRAPH control parameters 'X/LOGARITHMIC.' and 'Y/LOGARITHMIC.', which determine whether the X and Y axes are linear or logarithmic. Possible values are as follows:

  1 x axis linear, y axis linear
  2 x axis linear, y axis logarithmic
  3 x axis logarithmic, y axis linear
  4 x axis logarithmic, y axis logarithmic

  The default values of these parameters make both axes linear.

  If the parameters 'X/LOGARITHMIC.' and 'Y/LOGARITHMIC.' are reset by the routine DISPLA, they are given values which make them immune to being reset when 'SET.' = 2, 3, or 4 (see the discussion of the argument LSET, above).

In calls to the routines AGSETP, AGSETF, AGSETI AGSETC, AGGETP, AGGETF, AGGETI, and AGGETC:

- **TPGN** is a character string identifying a group of AUTOGRAPH control parameters. It is of the form 'K1/K2/K3/.../Kn.'. Each Ki is a keyword. The keyword K1 specifies a group of control parameters, K2 a subgroup of that group, K3 a subgroup of that subgroup, etc. See *AUTOGRAPH: A Graphing Utility* for a more complete description of these parameter-group names and the ways in which they may be abbreviated.
FURA is an array, from which control-parameter values are to be taken (the routine AGSETP) or into which they are to be stored (the routine AGGETP). Note that the array is real; all of the AUTOGRAPH parameters are stored internally as reals.

LURA is the length of the user array FURA.

FUSR is a variable, from which a single control parameter value is to be taken (the routine AGSETF) or in which it is to be returned (the routine AGGETF). Note that the variable is real.

IUSR is a variable, from which a single-control parameter value is to be taken (the routine AGSETI) or in which it is to be returned (the routine AGGETI). Note that, since the control parameters are stored internally as reals, each of the routines AGSETI and AGGETI does a conversion — from integer to real or vice-versa. Note also that AGSETI and AGGETI should only be used for parameters which have intrinsically integral values.

CUSR is a character variable from which a character string is to be taken (the routine AGSETC) or into which it is to be retrieved (the routine AGGETC). The control parameter affected by the call contains the floating-point equivalent of an integer identifier returned by the routine which stashes the character string and tendered to the routine which retrieves it (sort of the automated equivalent of a hat check). Note that AGSETC and AGGETC should only be used for parameters which intrinsically represent character strings.

In calls to the routine AGSTUP:

XDRA is an array of x coordinates of user data — usually, but not necessarily, the same data which will later be used in calls to the routine AGCURV.

NVIX is the number of vectors of data in XDRA — if XDRA is doubly-dimensioned, NVIX would normally have the value of its second dimension, if XDRA is singly-dimensioned, a 1.

IIVX is the index increment between vectors in XDRA — if XDRA is doubly-dimensioned, IIVX would normally have the value of its first dimension, if XDRA is singly-dimensioned, a dummy value.

NEVX is the number of elements in each data vector in XDRA — if XDRA is doubly-dimensioned, NEVX would normally have the value of its first dimension, if XDRA is singly-dimensioned, the value of that single dimension.

IIEX is the index increment between elements of a data vector in XDRA — normally a 1.

YDRA, NVIY, IIVY, NEVY, and IIEY are analogous to XDRA, NVIX, IIVX, NEVX, and IIEX, but define y-coordinate data.

In calls to the routine AGCURV:

XVEC is a vector of x coordinate data.
- IIEX is the index increment between elements in XVEC. AGCURV will use XVEC(1), XVEC(1+IIEX), XVEC(1+2*IIEX), etc.
- YVEC is a vector of y coordinate data.
- IIEY is the index increment between elements in YVEC. AGCURV will use YVEC(1), YVEC(1+IIEY), YVEC(1+2*IIEY), etc.
- NEXY is the number of points defining the curve to be drawn.
- KDSH is a dashed-line selector. Possible values are as follows:
  - If KDSH is zero, AUTOGRAPH will assume that the user has called the routine DASHD (see the section DASHCHAR in this manual) to define the dashed-line pattern to be used.
  - If KDSH is less than zero and has absolute value M, AUTOGRAPH will use the Mth (modulo 26) alphabetic dashed-line pattern. Each of these patterns defines a solid line interrupted every so often by a letter of the alphabet.
  - If KDSH is greater than zero and has the value M, AUTOGRAPH will use the Mth (modulo N) dashed-line pattern in the group of N dashed-line patterns defined by the AUTOGRAPH control parameters in the group named 'DASH/PATTERNS.'. The default values of these parameters specify solid lines.

In calls to the routines AGSAVE and AGRSTR:
- IFNO is the unit number associated with a file to which a single unformatted logical record of data is to be written, or from which such a record is to be read, by AUTOGRAPH. The file is not rewound before being written or read; positioning it properly is the user's responsibility.

In calls to the function AGBNCH:
- IDSH is a 16-bit binary dash pattern, the character equivalent of which is to be returned as the value of AGBNCH.

In calls to the function AGDSHN:
- IDSH is the number of the dash pattern parameter whose name is to be returned as the value of the function AGDSHN.

In calls to the routine AGUTOL:
- IAXS is the number of the axis. The values 1, 2, 3, and 4 imply the left, right, bottom, and top axes, respectively.
- FUNS is the value of the parameter 'AXIS/s/FUNCTION.' which may be used to select the desired mapping function for axis IAXS. It is recommended that the default value (zero) be used to specify the identity mapping. A non-zero value may be integral (1., 2., etc.) and serve purely to select the code to be executed or it may be the value of a real parameter in the equations defining the mapping.
* IDMA specifies the direction of the mapping. A value greater than zero indicates that VINP is a value in the user system and that VOTP is to be a value in the label system, a value less than zero the opposite.

* VINP is an input value in one coordinate system along the axis.

* VOTP is an output value in the other coordinate system along the axis.

In calls to the routine AGCHAX:

* IFLG is zero if a particular object is about to be drawn, non-zero if it has just been drawn.

* IAXS is the number of the axis being drawn. The values 1, 2, 3, and 4 indicate the left, right, bottom, and top axes, respectively.

* IPRT indicates the part of the axis being drawn. Possible values are as follows:
  1 implies the line of the axis.
  2 implies a major tick.
  3 implies a minor tick.
  4 implies the mantissa of a numeric label.
  5 implies the exponent of a numeric label.

* VILS is the value in the label system at the point where the part is being drawn. For IPRT = 1, VILS is zero.

In calls to the routine AGCHCU:

* IFLG is zero if a particular object is about to be drawn, non-zero if it has just been drawn.

* KDSH is the value with which AGCURV was called, as follows:

<table>
<thead>
<tr>
<th>AGCURV called by</th>
<th>Value of KDSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZY</td>
<td>1</td>
</tr>
<tr>
<td>EZXY</td>
<td>1</td>
</tr>
<tr>
<td>EZMY</td>
<td>&quot;n&quot; or &quot;-n&quot;, where n is the curve number</td>
</tr>
<tr>
<td>EZMXY</td>
<td>&quot;n&quot; or &quot;-n&quot;, where n is the curve number</td>
</tr>
<tr>
<td>the user program</td>
<td>the user value</td>
</tr>
</tbody>
</table>

In calls to the routine AGCHIL:

* IFLG is zero if a particular object is about to be drawn, non-zero if it has just been drawn.

* LBNM is a character variable containing the name of the label being drawn.

* LNNO is the number of the line being drawn.
In calls to the routine AGCHNL:

- IAXS is the number of the axis being drawn. The values 1, 2, 3, and 4 imply the left, right, bottom, and top axes, respectively.

- VILS is the value to be represented by the numeric label, in the label system for the axis. The value of VILS must not be altered.

- CHRM, on entry, is a character string containing the mantissa of the numeric label, as it will appear if AGCHNL makes no changes. If the numeric label includes a "times" symbol, it is represented by a blank in CHRM. (See IPXM, below.) CHRM may be modified.

- MCIM is the length of CHRM — the maximum number of characters that it will hold. The value of MCIM must not be altered.

- NCIM, on entry, is the number of meaningful characters in CHRM. If CHRM is changed, NCIM should be changed accordingly.

- IPXM, on entry, is zero if there is no "times" symbol in CHRM; if it is non-zero, it is the index of a character position in CHRM. If AGCHNL changes the position of the "times" symbol in CHRM, removes it, or adds it, the value of IPXM must be changed.

- CHRE, on entry, is a character string containing the exponent of the numeric label, as it will appear if AGCHNL makes no changes. CHRE may be modified.

- MCIE is the length of CHRE — the maximum number of characters that it will hold. The value of MCIE must not be altered.

- NCIE, on entry, is the number of meaningful characters in CHRE. If CHRE is changed, NCIE should be changed accordingly.
CONRAN

SUBROUTINE CONRAN (XD,YD,ZD,NDP,WK,IWK,SCRARR)

Standard and smooth versions of CONRAN

<table>
<thead>
<tr>
<th>Dimension of Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD(NDP), YD(NDP), ZD(NDP), WK(13*NDP)</td>
<td>Array of dimensions for input data</td>
</tr>
<tr>
<td>IWK((27+NCP)*NDP), SCRARR(RESOLUTION**2)</td>
<td>Work array and output resolution</td>
</tr>
</tbody>
</table>

where NCP = 4 and RESOLUTION = 40 by default.

Latest Revision August 1987

Overview CONRAN performs contouring of irregularly distributed data. It is the standard and smooth members of the CONRAN family. This version will plot contours; smooth them using splines under tension (if the package DASHSMTH is loaded); plot a perimeter or grid; title the plot; print a message giving the contour intervals below the map; plot the input data on the map; and label the contour lines.

Purpose CONRAN plots contour lines using random, sparse, or irregular data sets. The data is triangulated and then contoured. Contouring is performed using interpolation of the triangulated data. There are two methods of interpolation: C1 surfaces and linear.

Usage CALL CONRAN(XD,YD,ZD,NDP,WK,IWK,SCRARR)

An option setting routine can also be invoked, see writeup below. FRAME must be called by the user.

If different colors (or intensities) are to be used for normal intensity, low intensity, or text output, then the values in common block RANINT should be changed:

- IRANMJ Color index for normal (major) intensity lines.
- IRANMN Color index for low intensity lines.
- IRANTX Color index for text (labels).

Arguments

<table>
<thead>
<tr>
<th>On Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD</td>
<td>Array of dimension NDP containing the X-coordinates of the data points.</td>
</tr>
<tr>
<td>YD</td>
<td>Array of dimension NDP containing the Y-coordinates of the data points.</td>
</tr>
<tr>
<td>ZD</td>
<td>Array of dimension NDP containing the data values at the points.</td>
</tr>
<tr>
<td>NDP</td>
<td>Number of data points (must be 4 or greater) to be contoured.</td>
</tr>
<tr>
<td>WK</td>
<td>Real work array of dimension at least 13*NDP.</td>
</tr>
</tbody>
</table>
Integer work array. When using C1 surfaces the array must be at least \( IWK((27+NCP)\times NDP) \). When using linear interpolation the array must be at least \( IWK((27+4)\times NDP) \).

**SCRARR** Real work array of dimension at least \( (\text{RESOLUTION}^2) \) where \( \text{RESOLUTION} \) is described in the SSZ option below. \( \text{RESOLUTION} \) is 40 by default.

**On Output** All arguments remain unchanged except the scratch arrays \( IWK, WK, \) and \( \text{SCRARR} \), which have been written into. If making multiple runs on the same triangulation, \( IWK \) and \( WK \) must be saved and returned to the next invocation of \( \text{CONRAN} \).

**Entry Points**
- \( \text{CONRAN} \)
- \( \text{CONDDET} \)
- \( \text{CONINT} \)
- \( \text{CONCAL} \)
- \( \text{CONLOC} \)
- \( \text{CONTNG} \)
- \( \text{CONDRW} \)
- \( \text{CONCLS} \)
- \( \text{CONSTP} \)
- \( \text{CONBDN} \)
- \( \text{CONTLK} \)
- \( \text{CONOP1} \)
- \( \text{CONOP2} \)
- \( \text{CONOP3} \)
- \( \text{CONOP4} \)
- \( \text{CONXCH} \)
- \( \text{CONREO} \)
- \( \text{CONCOM} \)
- \( \text{CONSLD} \)
- \( \text{CONPMM} \)
- \( \text{CONGEN} \)
- \( \text{CONLCM} \)
- \( \text{CONOUT} \)
- \( \text{CONSSD} \)
- \( \text{CONRA1} \)
- \( \text{CONRA2} \)
- \( \text{CONRA3} \)
- \( \text{CONRA4} \)
- \( \text{CONRA5} \)
- \( \text{CONRA6} \)
- \( \text{CONRA7} \)
- \( \text{CONRA8} \)
- \( \text{CONRA9} \)
- \( \text{CONR10} \)
- \( \text{CONR11} \)
- \( \text{CONR12} \)
- \( \text{CONR13} \)
- \( \text{CONR14} \)
- \( \text{CONR15} \)
- \( \text{CONR16} \)
- \( \text{CONR17} \)
- \( \text{RANINT} \)
- \( \text{INTPR} \) from the DASH package

**I/O** Plots the contour map and, via the ERPRT77 package, outputs messages to the message output unit; at NCAR this unit is the printer. The option values are all listed on standard ERPRT77 output unit; at NCAR this unit is the printer.

**Precision** Single

**Required Library Routines**
- Standard version: DASHCHAR, which at NCAR is loaded by default.
- Smooth version: DASHSMTH which must be requested at NCAR. Both versions require CONCOM, CONTERP, GRIDAL, the ERPRT77 package, and the SPPS.

**Required GKS Level** 0A

**Language** FORTRAN 77

**Algorithm** The sparse data is triangulated and a virtual grid is laid over the triangulated area. Each virtual grid point receives an interpolated value. The grid is scanned once for each contour level and all contours at that level are plotted. There are two methods of interpolation. The first is a smooth data interpolation scheme based on Lawson's C1 surface interpolation algorithm, which has been refined by Hirosha Akima. Parts of Akima's algorithm are used in this package. See the "Reference" section below. The second is a linear interpolation scheme. When data is sparse
it is usually better to use the C1 interpolation. If you have dense data (over 100 points) then the linear interpolation will give the better results.

Portability
ANSI FORTRAN

Operation
CALL CONRAN (XD,YD,ZD,NDP,WK,IWK,SCRARR)

FRAME must be called by the user.

CONRAN has many options, each of which may be changed by calling one of the four subroutines CONOP1, CONOP2, CONOP3, or CONOP4. The number of arguments to each CONOP routine is the same as the final suffix character in the routine’s name.

The CONOP routines are called before CONRAN is called, and values set by these calls continue to be in effect until they are changed by another call to a CONOP routine.

All the CONOP routines have as their first argument a character string to identify the option being changed. This is the only argument to CONOP1. CONOP2 has an integer second argument. CONOP3 has a real array (or constant) as its second argument and an integer (usually the dimension of the array) as its third argument. CONOP4 has a character string as its second argument and integers for the third and fourth arguments.

Only the first two characters on each side of the equal sign are scanned. Therefore only 2 characters for each option are required on input to CONOP (i.e. ‘SCA=PRI’ and ‘SC=PR’ are equivalent).

Remember, there must be at least 4 data points. This is equal to the default number of data points to be used for estimation of partial derivatives at each data point. The estimated partial derivatives are used for the construction of the interpolating polynomial’s coefficients.

Listed below are options which can enhance your plot. An example of an appropriate CONOP call is given for each option. A complete list of default settings follows the last option.

Options

CHL

This flag determines how the high and low contour values are set. These contour values may be set by the program or by the user. If CHL=OFF, the program examines the user’s input data and determines both the high and low values. If CHL=ON, the user must specify the desired high (HI) and low (FLO) values. The default is CHL=OFF.

If program set: CALL CONOP3('CHL=OFF',0.,0)

If user set: CALL CONOP3('CHL=ON',ARRAY,2)
where ARRAY(1)=HI, ARRAY(2)=FLO
Note: The values supplied for contour increment and contour high and low values assume the unscaled data values. See the SDC flag, below.

Example: CALL CONOP3('CHL=ON',ARRAY,2)
where ARRAY(1)=5020. (the desired high contour value) and ARRAY(2)= 2000 (the desired low contour value). These are floating point numbers.

CIL
This flag determines how the contour increment (CINC) is set. The increment is either calculated by the program (CIL=OFF) using the range of high and low values from the user's input data, or set by the user (CIL=ON). The default is CIL=OFF.

If program set: CALL CONOP3('CIL=OFF',0.,0)
If user set: CALL CONOP3('CIL=ON',CINC,1)

Note: By default, the program will examine the user's input data and determine the contour interval (CINC) at some appropriate range between the level of high and low values supplied, usually generating between 15 and 20 contour levels.

Example: CALL CONOP3('CIL=ON',15.,1)
where 15. represents the contour increment desired by the user.

CON
This flag determines how the contour levels are set. If CON=ON, the user must specify the array of contour values and the number of contour levels. A maximum of 30 contour (NCL) levels is permitted. If CON=OFF, default values are used. In this case, the program will calculate the values for the array and NCL using input data. The default is OFF.

If program set: CALL CONOP3('CON=OFF',0.,0)
If user set: CALL CONOP3('CON=ON',ARRAY,NCL)

Note: The array (ARRAY) contains the contour levels (floating point only) and NCL is the number of levels. The maximum number of contour levels allowed is 30. When assigning the array of contour values, the values must be ordered from smallest to largest.

Example: DATA RLIST(1), . . . ,RLIST(5)/1.,2.,3.,10.,12./
CALL CONOP3('CON=ON',RLIST,5)
where 'RLIST' contains the user specified contour levels, and 5 is the number of user specified contour levels (NCL).
Warning on contour options: It is illegal to use the CON option when either CIL or CHL are activated. If this is done, the option call that detected the error will not be executed.

**DAS**
This flag determines which contours are represented by dashed lines. The user sets the dashed line pattern. The user may specify that dashed lines be used for contours whose value is less than, equal to, or greater than the dash pattern breakpoint (see the DBP option below), which is zero by default. If DAS=OFF (the default value), all solid lines are used.

- All solid lines: CALL CONOP4('DAS=OFF',' ',0,0)
- If greater: CALL CONOP4('DAS=GTR',PAT,0,0)
- If equal: CALL CONOP4('DAS=EQU',PAT,0,0)
- If less: CALL CONOP4('DAS=LSS',PAT,0,0)
- If all same: CALL CONOP4('DAS=ALL',PAT,0,0)

Note: PAT must be a ten character string with a dollar sign ($) for solid and a single quote (') for blank. Recall that in FORTRAN 77, in a quoted string a single quote is represented by two single quotes ("').

Example: CALL CONOP4('DAS=GTR','$$$$"$$$$',O,O)

**DBP**
This flag determines how the dash pattern break point (BP) is set. If DBP=ON, BP must be set by the user by specifying BP. If DBP=OFF the program will set BP to the default value which is zero.

- If program set: CALL CONOP3('DBP=OFF',0.,0)
- If user set: CALL CONOP3('DBP=ON',BP,1)

Note: BP is a floating point number where the break for GTR and LSS contour dash patterns are defined. BP is assumed to be given relative to the untransformed contours.

Example: CALL CONOP3('DBP=ON',5.,1)
where 5. is the user specified break point.

**DEF**
Reset flags to default values. Activating this option sets all flags to the default value. DEF has no 'ON' or 'OFF' states.

To activate: CALL CONOP1('DEF')

**EXT**
Flag to set extrapolation. Normally all CONRAN versions will only plot the boundaries of the convex hull defined by the user’s data. To have the contours fill the rectangular area of the frame, set the EXT switch ON. The default is OFF.

To turn on: CALL CONOP1('EXT=ON')
To turn off: CALL CONOP1('EXT=OFF')
**FMT** Flag for the format of the plotted input data values. If FMT=OFF, the default values for FT, L, and IF are used. The default values are:

- **FT** = '(G10.3)'
- **L** = 7 characters including the parentheses
- **IF** = 10 characters printed in the output field by the format

If FMT=ON, the user must specify values for FT, L, and IF. All user specified values must be given in the correct format.

If program set: CALL CONOP4('FMT=OFF'; ' ',0,0)

If user set: CALL CONOP4('FMT=ON',FT,L,IF)

Note: FT is a character string containing the format. The format must be enclosed in parentheses. Any format, up to 10 characters which is allowed at your installation will be accepted. L is the number of characters in FT. IF is the length of the field created by the format.

Example: CALL CONOP4('FMT=ON','(G30.2)',7,30)

**Warning:** CONRAN will not test for a valid format. The format is only allowed to be 10 characters long.

**GRI** Flag to display the grid. GRI is OFF by default.

To turn on: CALL CONOP1('GRI=ON')

To turn off: CALL CONOP1('GRI=OFF')

Note: If GRI is ON, the virtual grid will be superimposed over the contour plot. The X and Y tick intervals will be displayed under the map only if PER=ON (see PER).

**INT** Flag to determine the intensities of the contour lines and other parts of the plot. If INT=OFF, all intensities are set to the default values. If INT=ALL, all intensities are set to the given value, IVAL. If INT is set to one of the other possible options (MAJ, MIN, LAB or DAT), the intensity level for that option is set to the given value, IVAL.

- If program set: CALL CONOP2('INT=OFF',0)
- All the same: CALL CONOP2('INT=ALL',IVAL)
- Major lines: CALL CONOP2('INT=MAJ',IVAL)
- Minor lines: CALL CONOP2('INT=MIN',IVAL)
- Title and message: CALL CONOP2('INT=LAB',IVAL)
- Data values: CALL CONOP2('INT=DAT',IVAL)
Note: 'INT=DAT' relates to the plotted data values and the plotted maximums and minimums.

Note: IVAL is the intensity desired. IVAL values range from 0 to 255 or the character strings 'LO' and 'HI'. The default is 'HI' except for INT=MIN which is set to 'LO'.

Example: CALL CONOP2('INT=ALL','110')

ITP
Set the interpolation scheme. There are two schemes — C1 surfaces and linear. The C1 method takes longer but will give the best results when the data is sparse (less than 100 points). The linear method will produce a better plot when there is a dense data set. The default is C1 surface.

For C1 surface CALL CONOP1('ITP=C1')

For linear CALL CONOP1('ITP=LIN')

LAB
This flag can be set to either label the contours (LAB=ON) or not (LAB=OFF). The default value is LAB=ON.

To turn on: CALL CONOP1('LAB=ON')

To turn off: CALL CONOP1('LAB=OFF')

LOT
Flag to list options on the printer. The default value is set to OFF, and no options will be displayed.

To turn on: CALL CONOP1('LOT=ON')

To turn off: CALL CONOP1('LOT=OFF')

Note: If users want to print the option values, they should turn this option ON. The option values will be sent to the standard output unit as defined by the support routine I1MACH.

LSZ
This flag determines the label size. If LSZ=OFF, the default ISZLSZ value will be used. If LSZ=ON, the user should specify ISZLSZ. The default value is 9 plotter address units.

If program set: CALL CONOP2('LSZ=OFF',0)

If user set: CALL CONOP2('LSZ=ON',ISZLSZ)

Note: ISZLSZ is the requested character size in plotter address units.

Example: CALL CONOP2('LSZ=ON',4)
where 4 is the user desired integer plotter address units.
**MES** Flag to plot a message. The default is ON.

To turn on: CALL CONOP1('MES=ON')

To turn off: CALL CONOP1('MES=OFF')

Note: If MES=ON, a message is printed below the plot giving contour intervals and execution time in seconds. If PER or GRI is ON, the message also contains the X and Y tick intervals.

**NCP** Flag to indicate the number of data points used for the partial derivative estimation. If NCP=OFF, NUM is set to 4, which is the default value. If NCP=ON, the user must specify NUM greater than or equal to 2.

If program set: CALL CONOP2('NCP=OFF',0)

If user set: CALL CONOP2('NCP=ON',NUM)

Note: NUM = number of data points used for estimation. Changing this value effects the contours produced and the size of input array IWK.

Example: CALL CONOP2('NCP=ON',3)

**PDV** Flag to plot the input data values. The default value is PDV=OFF.

To turn on: CALL CONOP1('PDV=ON')

To turn off: CALL CONOP1('PDV=OFF')

Note: If PDV=ON, the input data values are plotted relative to their location on the contour map. If you only wish to see the locations and not the values, set PDV=ON and change FMT to produce an asterisk (*) such as (I1).

**PER** Flag to set the perimeter. The default value is PER=ON, which causes a perimeter to be drawn around the contour plot.

To turn on: CALL CONOP1('PER=ON')

To turn off: CALL CONOP1('PER=OFF')

Note: If MES is ON, the X and Y tick intervals will be given. These are the intervals in user coordinates that each tick mark represents.

**PMM** Flag to plot relative minimums and maximums. This flag is OFF by default.

To turn off: CALL CONOP1('PMM=OFF')

To turn on: CALL CONOP1('PMM=ON')
PSL  Flag which sets the plot shield option. The outline of the shield will be drawn on the same frame as the contour plot. By default this option is OFF (see SLD option).

Draw the shield: CALL CONOP1('PSL=ON')

Don't draw it: CALL CONOP1('PSL=OFF')

REP  Flag indicating the use of the same data in a new execution. The default value is OFF.

To turn on: CALL CONOP1('REP=ON')

To turn off: CALL CONOP1('REP=OFF')

Note: If REP=ON, the same X-Y data and triangulation are to be used, but it is assumed the user has changed contour values or resolution for this run. Scratch arrays WK and IWK must remain unchanged.

SCA  Flag for scaling of the plot on a frame. This flag is ON by default.

User scaling: CALL CONOP1('SCA=OFF')

Program scaling: CALL CONOP1('SCA=ON')

Prior window: CALL CONOP1('SCA=PRI')

Note: With SCA=OFF, plotting instructions will be issued using the user's input coordinates, unless they are transformed via FX and FY transformations. Users will find an extended discussion in the "Interfacing With Other Graphics Routines" section below. The SCA option assumes that all input data falls into the current window setting. With SCA=ON, the entry point will establish a viewport so that the user's plot will fit into the center 90 percent of the frame. When SCA=PRI, the program maps the user's plot instructions into the portion of the frame defined by the current normalization transformation. SCA=OFF should be used to interface with EZMAP.

SDC  Flag to determine how to scale the data on the contours. If SDC=OFF, the floating point value is given by scale. If SDC=ON, the user may specify SCALE. The default value for SCALE is 1.

If program set: CALL CONOP3('SDC=OFF',0.,0)

If user set: CALL CONOP3('SDC=ON',SCALE,1)

Note: The data plotted on contour lines and the data plotted for relative minimums and maximums will be scaled by the floating point value given by SCALE. Typical SCALE values are 10., 100., 1000., etc. The original data values are multiplied by
SCALE. SCALE must be a floating point number and is displayed in the message (see MES).

Example: CALL CONOP2('SDC=ON',100.,1)

SLD Activate or deactivate the shielding option. When this option is activated, only those contours within the shield are drawn. The shield is a polygon specified by the user which must be given in the same coordinate range as the data. It must define only one polygon.

To activate the shield:
   CALL CONOP3('SLD=ON',ARRAY,ICSD)

To deactivate the shield:
   CALL CONOP3('SLD=OFF',0.,0)

Note: ARRAY is a real array ICSD elements long. The first ICSD/2 elements are X coordinates and the second ICSD/2 elements are Y coordinates. ICSD is the length of an entire array, the number of (X + Y) shield coordinates. The polygon must be closed, that is the first and last points describing it must be the same.

Example: DIMENSION SHLD
   DATA SHLD/ 7.,10.,10.,7.,7.,7.,7.,10.,10.,7./
   CALL CONOP3 ('SLD=ON',SHLD,10)

SML Flag to determine the size of minimum and maximum contour labels. If SML=OFF, the ISZSML default value of 15 is used. If SML=ON, the user must specify ISZSML.

If program set: CALL CONOP2('SML=OFF',0)

If user set: CALL CONOP2('SML=ON',ISZSML)

Note: ISZSML is an integer number which is the size of labels in plotter address units as defined in the SPPS entry WTSTR.

Example: CALL CONOP2('SML=ON',12)

SPD Flag for the size of the plotted input data values. If SPD=OFF, the value of ISZSPD is 8, which is the default. If SPD=ON, the user must specify ISZSPD.

If program set: CALL CONOP2('SPD=OFF',0)

If user set: CALL CONOP2('SPD=ON',ISZSPD)

Note: ISZSPD is an integer number giving the size to plot the data values in plotter address units as defined in the SPPS entry WTSTR.
Example: CALL CONOP2('SPD=ON',6)

**SSZ**  
Flag to determine the resolution (number of steps in each direction). If SSZ=ON, the user sets ISTEP, or, if SSZ=OFF, the program will automatically set ISTEP at the default value of 40.

If program set: CALL CONOP2('SSZ=OFF',0)

If user set: CALL CONOP2('SSZ=ON',ISTEP)

Note: ISTEP is an integer specifying the density of the virtual grid. In most cases, the default value of 40 produces pleasing contours. For coarser but quicker contours, lower the value. For smoother contours at the expense of taking longer time, raise the value.

Note: For step sizes greater than 200 in CONRAN, the arrays PV in common CONRA1 and ITLOC in common CONRA9, must be expanded to about 10 more than ISTEP. See CONRA1 and CONRA9 comments below for more information.

Example: CALL CONOP2('SSZ=ON',25)  
This ISTEP value will produce a coarse contour.

**STL**  
Flag to determine the size of the title. ISZSTL may be set by the user (STL=ON), or the program will set it to the default size of 16 plotter address units (STL=OFF).

If program set: CALL CONOP2('STL=OFF',0)

If user set: CALL CONOP2('STL=ON',ISZSTL)

Note: When 30 or 40 characters are used for the title, the default size of 16 plotter address units works well. For longer titles, a smaller title size is required.

Example: CALL CONOP2('STL=ON',13)

**TEN**  
Flag to determine the tension factor applied when smoothing contour lines. The user may set TENS or allow the program to set the value. If user set, TENS must have a value greater than zero and less than or equal to 30. The default value is 2.5.

If program set: CALL CONOP3('TEN=OFF',0,0)

If user set: CALL CONOP3('TEN=ON',TENS,1)

Note: TENS is not available in the standard version of CONRAN. Smoothing of contour lines is accomplished with splines under tension. To adjust the amount of smoothing applied, adjust the tension factor. Setting TENS very large (i.e. 30.), effectively shuts off smoothing.
Example: CALL CONOP3('TEN=ON',14.,1)

**TFR**
Flag to advance the frame before triangulation. The default value is TFR=ON, which means that the contours and the triangles will be plotted on separate frames.

If program set: CALL CONOP1('TFR=ON')

To turn off: CALL CONOP1('TFR=OFF')

Note: Triangles are plotted after the contouring is completed. To see the triangles over the contours, turn this switch off.

**TLE**
Flag to place a title at the top of the plot. If TLE=ON, the user must specify CHARS and INUM. CHARS is the character string containing the title. INUM is the number of characters in CHARS. The default value is OFF.

To turn on: CALL CONOP4('TLE=ON',CHARS,INUM,0)

To turn off: CALL CONOP4('TLE=OFF',' ',0,0)

Note: If the title is longer than 64-characters, the character variable ISTRNG found in CONRA7 must be increased appropriately.

Example: CALL CONOP4('TLE=ON','VECTOR REVIEW',13,0)

**TOP**
Flag to plot only the triangles.

To turn off: CALL CONOP1('TOP=OFF')

To turn on: CALL CONOP1('TOP=ON')

Note: The user may wish to overlay the triangles on some other plot. 'TOP=ON' will allow that. This option when activated (TOP=ON), will set TRI=ON, and TFR=OFF. If the user wants TFR=ON, it should be set after TOP is set. If the user sets TOP=OFF it will set TRI=OFF and TFR=ON. If the user wants TRI or TFR different, set them after the TOP call.

**TRI**
Flag to plot the triangulation. The default is OFF and therefore the triangles are not drawn.

To turn on: CALL CONOP1('TRI=ON')

To turn off: CALL CONOP1('TRI=OFF')

Note: Plotting the triangles will indicate to the user where good and bad points of interpolation are occurring in the contour map. Equilateral triangles are optimal for interpolation. Quality degrades as triangles approach a long and narrow shape. The
convex hull of the triangulation is also a poor point of interpolation.

**Option Default Values**

Below are listed the default values for the various options given above. Unless the user specifies otherwise, these values will be used in execution of the various options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHL</td>
<td>OFF</td>
</tr>
<tr>
<td>CIL</td>
<td>OFF</td>
</tr>
<tr>
<td>CON</td>
<td>OFF</td>
</tr>
<tr>
<td>DAS</td>
<td>OFF</td>
</tr>
<tr>
<td>DBP</td>
<td>OFF</td>
</tr>
<tr>
<td>EXT</td>
<td>OFF</td>
</tr>
<tr>
<td>FMT</td>
<td>OFF</td>
</tr>
<tr>
<td>GRI</td>
<td>OFF</td>
</tr>
<tr>
<td>ITP-C1</td>
<td>OFF</td>
</tr>
<tr>
<td>LAB</td>
<td>ON</td>
</tr>
<tr>
<td>LOT</td>
<td>OFF</td>
</tr>
<tr>
<td>LSZ</td>
<td>OFF</td>
</tr>
<tr>
<td>MES</td>
<td>ON</td>
</tr>
<tr>
<td>NCP</td>
<td>OFF</td>
</tr>
<tr>
<td>PDV</td>
<td>OFF</td>
</tr>
<tr>
<td>PER</td>
<td>ON</td>
</tr>
<tr>
<td>PMM</td>
<td>OFF</td>
</tr>
<tr>
<td>REP</td>
<td>OFF</td>
</tr>
<tr>
<td>SCA</td>
<td>ON</td>
</tr>
<tr>
<td>SDC</td>
<td>OFF</td>
</tr>
<tr>
<td>SLD</td>
<td>OFF</td>
</tr>
<tr>
<td>SML</td>
<td>OFF</td>
</tr>
<tr>
<td>SPD</td>
<td>OFF</td>
</tr>
<tr>
<td>SSZ</td>
<td>OFF</td>
</tr>
<tr>
<td>STL</td>
<td>OFF</td>
</tr>
<tr>
<td>TFR</td>
<td>ON</td>
</tr>
<tr>
<td>TLE</td>
<td>OFF</td>
</tr>
<tr>
<td>TEN</td>
<td>OFF</td>
</tr>
<tr>
<td>TOP</td>
<td>OFF</td>
</tr>
<tr>
<td>TRI</td>
<td>OFF</td>
</tr>
</tbody>
</table>

The option default values given above, if used, will set default values for the following parameters:

**Default Values for User Specified Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>Up to 30 contour levels allowed. Values are computed by the program, based on input.</td>
</tr>
<tr>
<td>BP</td>
<td>0.</td>
</tr>
<tr>
<td>CINC</td>
<td>Computed by the program based on the range of HI and LO values of the input data.</td>
</tr>
<tr>
<td>FLO</td>
<td>Computed by the program based on the lowest unscaled input data.</td>
</tr>
<tr>
<td>FT</td>
<td>(G10.3) Parentheses must be included.</td>
</tr>
<tr>
<td>HI</td>
<td>Computed by the program based on the highest unscaled input data.</td>
</tr>
<tr>
<td>CHARS</td>
<td>No title</td>
</tr>
<tr>
<td>IF</td>
<td>10 characters</td>
</tr>
<tr>
<td>INUM</td>
<td>No title</td>
</tr>
<tr>
<td>IPAT</td>
<td>'$$$$$$$$$$' (This is a 10 character string.)</td>
</tr>
<tr>
<td>ISZLSZ</td>
<td>9 plotter address units</td>
</tr>
</tbody>
</table>
### Options That Affect the Contours

The shape of the contours may be modified by changing NCP and SSZ. NCP controls the number of data points to be used in the interpolation. Increasing NCP causes more of the surrounding data to influence the point of interpolation. Some datasets cause difficulty when trying to produce meaningful contours (triangles which are long and narrow). By modifying NCP a user can fine-tune a plot. Increasing ISTEP, the density of the virtual grid, will smooth out the contour lines and pick up more detail (new contours will appear as ISTEP increases and old ones will sometimes break into more distinct units). ISTEP is changed by the SSD option.

Note: If NCP.GT.25, arrays DSQO and IPC0 in CONDET must be adjusted accordingly. Also NCPSZ in CONBDN (25 by default), must be increased to NCP. The default value of NCP, which is 4, produces pleasing pictures in most cases. However, fine-tuning of the interpolation can be obtained by increasing the size of NCP, with a corresponding linear increase in work space.

The interpolation method used will also cause different looking contours. The C1 method is recommended when the data is sparse. It will smooth the data and add trends (false hills and valleys). The linear method is recommended when data is dense (GT 50 to 100) it will not smooth the data or add trends.
Normally the scaling factor will be set to OFF. In most cases mapping can be performed before calling the CONRAN entry point, thus saving the user from modifying the file. If reasonable results cannot be obtained, the statement functions, FX and FY, will have to be replaced. The routines having these statement functions are:

CONDRW, CONPDV, CONTLK, CONPMS, CONGEN

References


<table>
<thead>
<tr>
<th>CONRAN Error Messages</th>
<th>Error</th>
<th>Routine</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONRAN</td>
<td>INPUT PARAMETER NDP LT NCP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CONRAN</td>
<td>NCP GT MAX SIZE OR LT 2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CONTNG</td>
<td>ALL COLINEAR DATA POINTS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CONTNG</td>
<td>IDENTICAL INPUT DATA POINTS FOUND</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>CONOP</td>
<td>UNDEFINED OPTION</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CONCLS</td>
<td>CONSTANT INPUT FIELD</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CONOP</td>
<td>INCORRECT CONOP CALL USED</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CONOP</td>
<td>ILLEGAL USE OF CON OPTION WITH CIL OR CHL OPTIONS</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>CONOP</td>
<td>NUMBER OF CONTOUR LEVELS EXCEEDS 30</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CONDRW</td>
<td>CONTOUR STORAGE EXHAUSTED</td>
<td></td>
</tr>
</tbody>
</table>

This error is trapped and nullified by CONRAN. It serves to signal the user that a contour level may not be complete.

| 11                    | CONSTD| ASPECT RATIO OF X AND Y GREATER THAN 5 TO 1. This error may cause a poor quality plot. Usually this can be fixed by multiplying X or Y by a constant factor. If this solution is unacceptable, then increasing SSZ to a very large value may help. Note: This can be expensive. |
The errors listed above are defined as recoverable errors should the user wish to use them in that fashion. The documentation on the ERPRT77 package explains how to recover from an error.
CONRAQ
CONRAQ

SUBROUTINE CONRAQ (XD,YD,ZD,NDP,WK,IWK)
Quick version of CONRAN

Dimension of Arguments
XD(NDP), YD(NDP), ZD(NDP), WK(13*NDP)
IWK((27+NCP)*NDP), where NCP=4.

Latest Revision
August 1987

Overview
CONRAQ performs contouring of irregularly distributed data. It is the fastest member of the CONRAN family. This version will plot contours from your data. It has the ability to draw a perimeter or grid; title the plot; print a message giving the contour intervals below the map; and plot the input data on the map. This version is well suited for testing your data. Besides being the fastest version, it is also the smallest.

Purpose
CONRAQ plots contour lines using random, sparse, or irregular data sets. The data is triangulated and then contoured. Contouring is performed using interpolation of the triangulated data. There are two methods of interpolation: C1 surfaces and linear.

Usage
CALL CONRAQ(XD,YD,ZD,NDP,WK,IWK,)
An option setting routine can also be invoked; see the description below.
FRAME must be called by the user.

If different colors (or intensities) are to be used for normal intensity, low intensity, or text output then the values in common block RAQINT should be changed:

IRAQMJ  Color index for normal (major) intensity lines.
IRAQMN  Color index for low intensity lines
IRAQTX  Color index for text (labels)

ARGUMENTS

On Input
XD  Array of dimension NDP containing the X-coordinates of the data points.

YD  Array of dimension NDP containing the Y-coordinates of the data points.

ZD  Array of dimension NDP containing the data values at the points.

NDP  Number of data points (must be 4 or greater) to be contoured.

WK  Real work array of dimension at least 13*NDP.

IWK  Integer work array. When using C1 surfaces the array must be at least IWK((27+NCP)*NDP). When using linear interpolation the array must be at least IWK((27+4)*NDP).
On Output
All arguments remain unchanged except the scratch arrays IWK and WK, which have been written into. If making multiple runs on the same triangulation, IWK and WK must be saved and returned to the next invocation of CONRAQ.

Entry Points
CONRAQ, CONDET, CONINT, CONCAL, CONLOC, CONTNG, CONDRW, CONCLS, CONBDN, CONTOR, CONPDV, CONOP1, CONOP2, CONOP3, CONOP4, CONXCH, CONOUT, CONOT2, CONLIN

Common Blocks
CONRA1, CONRA2, CONRA3, CONRA4, CONRA5, CONRA6, CONRA7, CONRA8, CONRA9, CONR10, CONR11, CONR12, CONR14, CONR15, CONR16, CONR17, RAQINT

INTPR from the DASH package

I/O
Plots the contour map and, via the ERPRT77 package, outputs messages to the message output unit; at NCAR this unit is the printer. The option values are all listed on standard ERPRT77 output unit; at NCAR this unit is the printer.

Precision
Single

Required Library Routines
GRIDAL, CONCOM, CONTERP, the ERPRT77 package, and the SPPS.

Required GKS Level
0A

Note for NCAR Users
This routine is NOT part of the default libraries at NCAR. CONRAS must be acquired, compiled, and loaded to be used at NCAR.

Language
FORTRAN77

Algorithm
The sparse data is triangulated and a virtual grid is laid over the triangulated area. Each virtual grid point receives an interpolated value. The grid is scanned once for each contour level and all contours at that level are plotted. There are two methods of interpolation. The first is a smooth data interpolation scheme based on Lawson's C1 surface interpolation algorithm, which has been refined by Hirosha Akima. Parts of Akima's algorithm are used in this package. See the "REFERENCE" section below. The second is a linear interpolation scheme. When data is sparse, it is usually better to use the C1 interpolation. If you have dense data (over 100 points) then the linear interpolation will give the better results.

Portability
ANSI FORTRAN
Operation

CALL CONRAQ (XD,YD,ZD,NDP,WK,IWK)

FRAME must be called by the user.

CONRAQ has many options, each of which may be changed by calling one of the four subroutines CONOP1, CONOP2, CONOP3, or CONOP4. The number of arguments to each CONOP routine is the same as the final suffix character in the routine's name.

The CONOP routines are called before CONRAQ is called, and values set by these calls continue to be in effect until they are changed by another call to a CONOP routine.

All the CONOP routines have as their first argument a character string to identify the option being changed. This is the only argument to CONOP1. CONOP2 has an integer second argument. CONOP3 has a real array (or constant) as its second argument and an integer (usually the dimension of the array) as its third argument. CONOP4 has a character string as its second argument and integers for the third and fourth arguments.

Only the first two characters on each side of the equal sign are scanned. Therefore only 2 characters for each option are required on input to CONOP (i.e. 'SCA=PRI' and 'SC=PR' are equivalent.)

Remember, there must be at least 4 data points. This is equal to the default number of data points to be used for estimation of partial derivatives at each data point. The estimated partial derivatives are used for the construction of the interpolating polynomial's coefficients.

Listed below are options which can enhance your plot. An example of an appropriate CONOP call is given for each option. A complete list of default settings follows the last option.

Options

CHL

This flag determines how the high and low contour values are set. These contour values may be set by the program or by the user. If CHL=OFF, the program examines the user's input data and determines both the high and low values. If CHL=ON, the user must specify the desired high (HI) and low (FLO) values. The default is CHL=OFF.

If program set: CALL CONOP3('CHL=OFF',0.,0)

If user set: CALL CONOP3('CHL=ON',ARRAY,2)
where ARRAY(1)=HI, ARRAY(2)=FLO

Example: CALL CONOP3('CHL=ON',ARRAY,2)
where ARRAY(1)=5020, (the desired high contour value) and ARRAY(2)=2000 (the desired low contour value).
These are floating point numbers.
This flag determines how the contour increment (CINC) is set. The increment is either calculated by the program (CIL=OFF) using the range of high and low values from the user's input data, or set by the user (CIL=ON). The default is CIL=OFF.

If program set: CALL CONOP3('CIL=OFF',0.,0)

If user set: CALL CONOP3('CIL=ON',CINC,1)

Note: By default, the program will examine the user's input data and determine the contour interval (CINC) at some appropriate range between the level of high and low values supplied, usually generating between 15 and 20 contour levels.

Example: CALL CONOP3('CIL=ON',15.,1) where 15. represents the contour increment desired by the user.

This flag determines how the contour levels are set. If CON=ON, the user must specify the array of contour values and the number of contour levels. A maximum of 30 contour (NCL) levels is permitted. If CON=OFF, default values are used. In this case, the program will calculate the values for the array and NCL using input data. The default is OFF.

If program set: CALL CONOP3('CON=OFF',0.,0)

If user set: CALL CONOP3('CON=ON',ARRAY,NCL)

Note: The array (ARRAY) contains the contour levels (floating point only) and NCL is the number of levels. The maximum number of contour levels allowed is 30. When assigning the array of contour values, the values must be ordered from smallest to largest.

Example: DATA RLIST(1),...,RLIST(5)/1.,2.,3.,10.,12./

CALL CONOP3('CON=ON',RLIST,5) where 'RLIST' contains the user specified contour levels, and 5 is the number of user specified contour levels (NCL).

Warning on contour options: It is illegal to use the CON option when either CIL or CHL are activated. If this is done, the option call that detected the error will not be executed.

Reset flags to default values. Activating this option sets all flags to the default value. DEF has no 'ON' or 'OFF' states.

To activate: CALL CONOP1('DEF')

Flag to set extrapolation. Normally all CONRAN versions will only plot the boundaries of the convex hull defined by the user's
data. To have the contours fill the rectangular area of the frame, set the EXT switch ON. The default is OFF.

To turn on: CALL CONOP1('EXT=ON')

To turn off: CALL CONOP1('EXT=OFF')

**FMT**

Flag for the format of the plotted input data values. If FMT=OFF, the default values for FT, L, and IF are used. The default values are:

FT = '{(G10.3)'}
L = 7 characters including the parentheses
IF = 10 characters printed in the output field by the format

If FMT=ON, the user must specify values for FT, L, and IF. All user specified values must be given in the correct format.

If program set: CALL CONOP4('FMT=OFF', ' ', 0, 0)

If user set: CALL CONOP4('FMT=ON', FT, L, IF)

Note: FT is a character string containing the format. The format must be enclosed in parentheses. Any format, up to 10 characters which is allowed at your installation will be accepted. L is the number of characters in FT. IF is the length of the field created by the format.

Example: CALL CONOP4('FMT=ON','(G30.2)', 7, 30)

**Warning:** CONRAQ will not test for a valid format. The format is only allowed to be 10 characters long.

**GRI**

Flag to display the grid. GRI is OFF by default.

To turn on: CALL CONOP1('GRI=ON')

To turn off: CALL CONOP1('GRI=OFF')

Note: If GRI is ON, the virtual grid will be superimposed over the contour plot. The X and Y tick intervals will be displayed under the map only if PER=ON (see PER).

**INT**

Flag to determine the intensities of the contour lines and other parts of the plot. If INT=OFF, all intensities are set to the default values. If INT=ALL, all intensities are set to the given value, IVAL. If INT is set to one of the other possible options (MAJ, MIN, LAB or DAT), the intensity level for that option is set to the given value, IVAL.

If program set: CALL CONOP2('INT=OFF', 0)
All the same: CALL CONOP2('INT=ALL',IVAL)

Major lines: CALL CONOP2('INT=MAJ',IVAL)

Minor lines: CALL CONOP2('INT=MIN',IVAL)

Title and message:
CALL CONOP2('INT=LAB',IVAL)

Data values: CALL CONOP2('INT=DAT',IVAL)

Note: 'INT=DAT' relates to the plotted data values and the plotted maximums and minimums.

Note: IVAL is the intensity desired. IVAL values range from 0 to 255 or the character strings 'LO' and 'HI'. The default is 'HI' except for INT=MIN, which is set to 'LO'.

Example: CALL CONOP2('INT=ALL',110)

ITP

Set the interpolation scheme. There are two schemes — C1 surfaces and linear. The C1 method takes longer but will give the best results when the data is sparse (less than 100 points). The linear method will produce a better plot when there is a dense data set. The default is C1 surface.

For C1 surface CALL CONOP1('ITP=C1')

For linear CALL CONOP1('ITP=LIN')

LOT

Flag to list options on the printer. The default value is set to OFF, and no options will be displayed.

To turn on: CALL CONOP1('LOT=ON')

To turn off: CALL CONOP1('LOT=OFF')

Note: If users want to print the option values, they should turn this option ON. The option values will be sent to the standard output unit as defined by the support routine I1MACH.

MES

Flag to plot a message. The default is ON.

To turn on: CALL CONOP1('MES=ON')

To turn off: CALL CONOP1('MES=OFF')

Note: If MES=ON, a message is printed below the plot giving contour intervals and execution time in seconds. If PER or GRI is ON, the message also contains the X and Y tick intervals.
CONRAQ

NCP Flag to indicate the number of data points used for the partial derivative estimation. If NCP=OFF, NUM is set to 4, which is the default value. If NCP=ON, the user must specify NUM greater than or equal to 2.

If program set: CALL CONOP2('NCP=OFF',0)

If user set: CALL CONOP2('NCP=ON',NUM)

Note: NUM = number of data points used for estimation. Changing this value effects the contours produced and the size of input array IWK.

Example: CALL CONOP2('NCP=ON',3)

PDV Flag to plot the input data values. The default value is PDV=OFF.

To turn on: CALL CONOP1('PDV=ON')

To turn off: CALL CONOP1('PDV=OFF')

Note: If PDV=ON, the input data values are plotted relative to their location on the contour map. If you only wish to see the locations and not the values, set PDV=ON and change FMT to produce an asterisk (*) such as (I1).

PER Flag to set the perimeter. The default value is PER=ON, which causes a perimeter to be drawn around the contour plot.

To turn on: CALL CONOP1('PER=ON')

To turn off: CALL CONOP1('PER=OFF')

Note: If MES is ON, the X and Y tick intervals will be given. These are the intervals in user coordinates that each tick mark represents.

REP Flag indicating the use of the same data in a new execution. The default value is OFF.

To turn on: CALL CONOP1('REP=ON')

To turn off: CALL CONOP1('REP=OFF')

Note: If REP=ON, the same X-Y data and triangulation are to be used but it is assumed the user has changed contour values or resolution for this run. Scratch arrays WK and IWK must remain unchanged.
SCA  Flag for scaling of the plot on a frame. This flag is ON by default.

User scaling:  CALL CONOP1('SCA=OFF')

Program scaling: CALL CONOP1('SCA=ON')

Prior window: CALL CONOP1('SCA=PRI')

Note: With SCA=OFF, plotting instructions will be issued using the user's input coordinates, unless they are transformed via FX and FY transformations. Users will find an extended discussion in the "INTERFACING WITH OTHER GRAPHICS ROUTINES" section below. The SCA option assumes that all input data falls into the current window setting. With SCA=ON, the entry point will establish a viewport so that the user's plot will fit into the center 90 percent of the frame. When SCA=PRI, the program maps the user's plot instructions into the portion of the frame defined by the current normalization transformation. SCA=OFF should be used to interface with EZMAP.

SPD  Flag for the size of the plotted input data values. If SPD=OFF, the value of ISZSPD is 8, which is the default. If SPD=ON, the user must specify ISZSPD.

If program set:  CALL CONOP2('SPD=OFF',0)

If user set:  CALL CONOP2('SPD=ON',ISZSPD)

Note: ISZSPD is an integer number giving the size to plot the data values in plotter address units as defined in the SPPS entry WTSTR.

Example: CALL CONOP2('SPD=ON',6)

SSZ  Flag to determine the resolution (number of steps in each direction). If SSZ=ON, the user sets ISTEP, or, if SSZ=OFF, the program will automatically set ISTEP at the default value of 40.

If program set:  CALL CONOP2('SSZ=OFF',0)

If user set:  CALL CONOP2('SSZ=ON',ISTEP)

Note: ISTEP is an integer specifying the density of the virtual grid. In most cases, the default value of 40 produces pleasing contours. For coarser but quicker contours, lower the value. For smoother contours at the expense of taking longer time, raise the value.

Note: For step sizes greater than 200 in CONRAQ, the arrays PV in common CONRA1 and ITLOC in common CONRA9, must
be expanded to about 10 more than ISTEP. See CONRA1 and CONRA9 comments below for more information.

Example: CALL CONOP2('SSZ=ON',25) This ISTEP value will produce a coarse contour.

### STL
Flag to determine the size of the title. ISZSTL may be set by the user (STL=ON), or the program will set it to the default size of 16 plotter address units (STL=OFF).

If program set: CALL CONOP2('STL=OFF',0)

If user set: CALL CONOP2('STL=ON',ISZSTL)

Note: When 30 or 40 characters are used for the title, the default size of 16 plotter address units works well. For longer titles, a smaller title size is required.

Example: CALL CONOP2('STL=ON',13)

### TFR
Flag to advance the frame before triangulation. The default value is TFR=ON, which means that the contours and the triangles will be plotted on separate frames.

If program set: CALL CONOPI('TFR=ON')

To turn off: CALL CONOP1('TFR=OFF')

Note: Triangles are plotted after the contouring is completed. To see the triangles over the contours, turn this switch off.

### TLE
Flag to place a title at the top of the plot. If TLE=ON, the user must specify CHARS and INUM. CHARS is the character string containing the title. INUM is the number of characters in CHARS. The default value is OFF.

To turn on: CALL CONOP4('TLE=ON',CHARS,INUM,0)

To turn off: CALL CONOP4('TLE=OFF',' ',0,0)

Note: If longer than 64-character titles are desired, the character variable ISTRNG found in CONRA7 must be increased appropriately.

Example: CALL CONOP4('TLE=ON','VECTOR REVIEW' ,13,0)

### TOP
Flag to plot only the triangles.

To turn off: CALL CONOP1('TOP=OFF')
To turn on: CALL CONOP1('TOP=ON')

Note: The user may wish to overlay the triangles on some other plot. 'TOP=ON' will allow that. This option when activated (TOP=ON), will set TRI=ON, and TFR=OFF. If the user wants TFR=ON, it should be set after TOP is set. If the user sets TOP=OFF it will set TRI=OFF and TFR=ON. If the user wants TRI or TFR different, set them after the TOP call.

**TRI**
Flag to plot the triangulation. The default is OFF and therefore the triangles are not drawn.

To turn on: CALL CONOP1('TRI=ON')

To turn off: CALL CONOP1('TRI=OFF')

Note: Plotting the triangles will indicate to the user where good and bad points of interpolation are occurring in the contour map. Equilateral triangles are optimal for interpolation. Quality degrades as triangles approach a long and narrow shape. The convex hull of the triangulation is also a poor point of interpolation.

### Option Default Values
Below are listed the default values for the various options given above. Unless the user specifies otherwise, these values will be used in execution of the various options.

- CHL=OFF
- CIL=OFF
- CON=OFF
- EXT=OFF
- FMT=OFF
- GRI=OFF
- LOT=OFF
- MES=ON
- NCP=OFF
- PDV=OFF
- PER=ON
- REP=OFF
- SCA=ON
- SPD=OFF
- SSZ=OFF
- STL=OFF
- TLE=OFF
- TOP=OFF
- TRI=OFF
- ITP=C1

### Default Values for User Specified Parameters
The option default values given above, if used, will set default values for the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>Up to 30 contour levels allowed. Values are computed by the program, based on input.</td>
</tr>
<tr>
<td>CINC</td>
<td>Computed by the program based on the range of HI and LO values of the input data.</td>
</tr>
<tr>
<td>FLO</td>
<td>Computed by the program based on the lowest unscaled input data.</td>
</tr>
<tr>
<td>FT</td>
<td>(G10.3) Parentheses must be included.</td>
</tr>
</tbody>
</table>
### Parameter | Default
---|---
HI | Computed by the program based on the highest unscaled input data.
CHARS | No title
IF | 10 characters
INUM | No title
ISZSPD | 8 plotter address units
ISZSTL | 16 plotter address units
ISTEP | 40
IVAL | 'HI' for all except minor contour lines, which are 'LO'.
L | 7 characters (including both parentheses)
NCL | Computed by the program based on input data. Up to 30 contour levels are permitted.
NUM | 4 data points
SCALE | 1. (no scaling performed)

#### Options That Affect the Contours

The shape of the contours may be modified by changing NCP and SSZ. NCP controls the number of data points to be used in the interpolation. Increasing NCP causes more of the surrounding data to influence the point of interpolation. Some datasets cause difficulty when trying to produce meaningful contours (triangles that are long and narrow). By modifying NCP a user can fine-tune a plot. Increasing ISTEP, the density of the virtual grid, will smooth out the contour lines and pick up more detail (new contours will appear as ISTEP increases and old ones will sometimes break into more distinct units). ISTEP is changed by the SSD option.

Note: If NCP.GT.25, arrays DSQ0 and IPC0 in CONDET must be adjusted accordingly. Also NCPSZ in CONBDN (25 by default), must be increased to NCP. The default value of NCP, which is 4, produces pleasing pictures in most cases. However, fine-tuning of the interpolation can be obtained by increasing the size of NCP, with a corresponding linear increase in work space.

The interpolation method used will also cause different-looking contours. The C1 method is recommended when the data is sparse. It will smooth the data and add trends (false hills and valleys). The linear method is recommended when data is dense (GT 50 to 100); it will not smooth the data or add trends.

August 1987
Normally the scaling factor will be set to OFF. In most cases mapping can be performed before calling the CONRAQ entry point, thus saving the user from modifying the file. If reasonable results cannot be obtained, the statement functions, FX and FY, will have to be replaced. The routines having these statement functions are:

CONDRW, CONPDV, CONTLK, CONPMS, CONGEN


**CONRAQ Error Messages**

<table>
<thead>
<tr>
<th>Error</th>
<th>Routine</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONRAQ</td>
<td>INPUT PARAMETER NDP LT NCP</td>
</tr>
<tr>
<td>2</td>
<td>CONRAQ</td>
<td>NCP GT MAX SIZE OR LT 2</td>
</tr>
<tr>
<td>3</td>
<td>CONTNG</td>
<td>ALL COLINEAR DATA POINTS</td>
</tr>
<tr>
<td>4</td>
<td>CONTNG</td>
<td>IDENTICAL INPUT DATA POINTS FOUND</td>
</tr>
<tr>
<td>5</td>
<td>CONOP</td>
<td>UNDEFINED OPTION</td>
</tr>
<tr>
<td>6</td>
<td>CONCLS</td>
<td>CONSTANT INPUT FIELD</td>
</tr>
<tr>
<td>7</td>
<td>CONOP</td>
<td>INCORRECT CONOP CALL USED</td>
</tr>
<tr>
<td>8</td>
<td>CONOP</td>
<td>ILLEGAL USE OF CON OPTION WITH CIL OR CHL OPTIONS</td>
</tr>
<tr>
<td>9</td>
<td>CONOP</td>
<td>NUMBER OF CONTOUR LEVELS EXCEEDS 30</td>
</tr>
<tr>
<td>10</td>
<td>CONDRW</td>
<td>CONTOUR STORAGE EXHAUSTED This error is trapped and nullified by CONRAN. It serves to signal the user that a contour level may not be complete.</td>
</tr>
<tr>
<td>11</td>
<td>CONSTP</td>
<td>ASPECT RATIO OF X AND Y GREATER THAN 5 TO 1. This error may cause a poor quality plot. Usually this can be fixed by multiplying X or Y by a constant factor. If this solution is unacceptable then increasing SSZ to a very large value may help. Note: This can be expensive.</td>
</tr>
</tbody>
</table>

The errors listed above are defined as recoverable errors should the user wish to use them in that fashion. The documentation on the ERPRT77 package explains how to recover from an error.
CONRAS
CONRAS

SUBROUTINE CONRAS (XD,YD,ZD,NDP,WK,IWK,SCRARR)

Super version of CONRAN

Dimension of Arguments
XD(NDP), YD(NDP), ZD(NDP), WK(13*NDP)
IWK((27+NCP)*NDP), SCRARR(RESOLUTION**2)
where NCP = 4 and RESOLUTION = 40 by default.

Latest Revision
August 1987

Overview
CONRAS performs contouring of irregularly distributed data. It is the super member of the CONRAN family. This version will plot contours; smooth them using splines under tension; plot a perimeter or grid; title the plot; print a message giving the contour intervals below the map; plot the input data on the map; label the contour lines and eliminate crowding of contour lines. It is the biggest and slowest member of the family; it is intended to produce publication-quality maps.

Purpose
CONRAS plots contour lines using random, sparse, or irregular data sets. The data is triangulated and then contoured. Contouring is performed using interpolation of the triangulated data. There are two methods of interpolation: C1 surfaces and linear.

Usage
CALL CONRAS(XD,YD,ZD,NDP,WK,IWK,SCRARR)

An option setting routine can also be invoked, see writeup below. FRAME must be called by the user.

If different colors (or intensities) are to be used for normal intensity, low intensity, or text output then the values in common block RASINT should be changed:

IRASMJ Color index for normal (major) intensity lines.
IRASMN Color index for low intensity lines
IRASTX Color index for text (labels)

ARGUMENTS

On Input
XD Array of dimension NDP containing the X-coordinates of the data points.
YD Array of dimension NDP containing the Y-coordinates of the data points.
ZD Array of dimension NDP containing the data values at the points.
NDP Number of data points (must be 4 or greater) to be contoured.
WK Real work array of dimension at least 13*NDP
Integer work array. When using C1 surfaces the array must be at least \( IWK((27+NCP)\times NDP) \). When using linear interpolation the array must be at least \( IWK((27+4)\times NDP) \).

Real work array of dimension at least \( (RESOLUTION^2) \) where \( RESOLUTION \) is described in the SSZ option below. \( RESOLUTION \) is 40 by default.

On Output
All arguments remain unchanged except the scratch arrays \( IWK, WK, \) and \( SCRARR \), which have been written into. If making multiple runs on the same triangulation, \( IWK \) and \( WK \) must be saved and returned to the next invocation of \( CONRAS \).

Entry Points
\( CONRAS, \) \( CONDET, \) \( CONINT, \) \( CONCAL, \) \( CONLOC, \) \( CONTNG, \) \( CONDRW, \) \( CONCLS, \) \( CONTP, \) \( CONBDN, \) \( CONTLK, \) \( CONPDV, \) \( CONOP1, \) \( CONOP2, \) \( CONOP3, \) \( CONOP4, \) \( CONXCH, \) \( CONREO, \) \( CONCOM, \) \( CONCLD, \) \( CONPMS, \) \( CONSTP, \) \( CONGEN, \) \( CONLDM, \) \( CONEDC, \) \( CONOUT, \) \( CONOT2, \) \( CONCLD, \) \( CONSLD, \) \( CONLDM, \) \( CONLIN, \) \( CONSD, \) \( CONSSD \)

Common Blocks
\( CONRA1, \) \( CONRA2, \) \( CONRA3, \) \( CONRA4, \) \( CONRA5, \) \( CONRA6, \) \( CONRA7, \) \( CONRA8, \) \( CONRA9, \) \( CONR10, \) \( CONR11, \) \( CONR12, \) \( CONR13, \) \( CONR14, \) \( CONR15, \) \( CONR16, \) \( CONR17, \) \( RASINT \)

I/O
Plots the contour map and, via the ERPRT77 package, outputs messages to the message output unit; at NCAR this unit is the printer. The option values are all listed on standard ERPRT77 output unit; at NCAR this unit is the printer.

Precision
Single

Required Library Routines
\( DASHSUPR, \) \( GRIDAL, \) \( CONCOM, \) \( CONTERP, \) the ERPRT77 package, and the SPPS.

Required GKS Level
0A

Note for NCAR Users
This routine is NOT part of the default libraries at NCAR. \( CONRAS \) must be be acquired, compiled, and loaded to be used at NCAR.

Language
FORTRAN 77

Algorithm
The sparse data is triangulated, and a virtual grid is laid over the triangulated area. Each virtual grid point receives an interpolated value. The grid is scanned once for each contour level, and all contours at that level are plotted. There are two methods of interpolation. The first is a smooth data interpolation scheme based on Lawson's C1 surface interpolation algorithm, which has been refined by Hirosha Akima. Parts of Akima's algorithm are used in this package. See the "REFERENCE" section below. The second is a linear interpolation scheme. When data is
sparse, it is usually better to use the $C^1$ interpolation. If you have dense data (over 100 points), then the linear interpolation will give the better results.

**Portability**

ANSI FORTRAN

**Operation**

CALL CONRAS (XD,YD,ZD,NDP,WK,IWK,SCRARR)

FRAME must be called by the user.

CONRAS has many options, each of which may be changed by calling one of the four subroutines CONOP1, CONOP2, CONOP3, or CONOP4. The number of arguments to each CONOP routine is the same as the final suffix character in the routine’s name.

The CONOP routines are called before CONRAS is called, and values set by these calls continue to be in effect until they are changed by another call to a CONOP routine.

All the CONOP routines have as their first argument a character string to identify the option being changed. This is the only argument to CONOP1. CONOP2 has an integer second argument. CONOP3 has a real array (or constant) as its second argument and an integer (usually the dimension of the array) as its third argument. CONOP4 has a character string as its second argument and integers for the third and fourth arguments.

Only the first two characters on each side of the equal sign are scanned. Therefore only 2 characters for each option are required on input to CONOP (i.e. 'SCA=PRI' and 'SC=PR' are equivalent).

Remember, there must be at least 4 data points. This is equal to the default number of data points to be used for estimation of partial derivatives at each data point. The estimated partial derivatives are used for the construction of the interpolating polynomial’s coefficients.

Listed below are options that can enhance your plot. An example of an appropriate CONOP call is given for each option. A complete list of default settings follows the last option.

**Options**

**CHL**

This flag determines how the high and low contour values are set. These contour values may be set by the program or by the user. If CHL=OFF, the program examines the user’s input data and determines both the high and low values. If CHL=ON, the user must specify the desired high (HI) and low (FLO) values. The default is CHL=OFF.

If program set: CALL CONOP3('CHL=OFF',0,0)

If user set: CALL CONOP3('CHL=ON',ARRAY,2)
where ARRAY(1)=HI, ARRAY(2)=FLO
Note: The values supplied for contour increment and contour high and low values assume the unscaled data values. See the SDC flag, below.

Example: CALL CONOP3('CHL=ON',ARRAY,2) where ARRAY(1)=5020. (the desired high contour value) and ARRAY(2)=2000 (the desired low contour value). These are floating point numbers.

CIL This flag determines how the contour increment (CINC) is set. The increment is either calculated by the program (CIL=OFF) using the range of high and low values from the user's input data, or set by the user (CIL=ON). The default is CIL=OFF.

If program set: CALL CONOP3('CIL=OFF',0.,0)
If user set: CALL CONOP3('CIL=ON',CINC,1)

Note: By default, the program will examine the user's input data and determine the contour interval (CINC) at some appropriate range between the level of high and low values supplied, usually generating between 15 and 20 contour levels.

Example: CALL CONOP3('CIL=ON',15.,1) where 15. represents the contour increment desired by the user.

CON This flag determines how the contour levels are set. If CON=ON, the user must specify the array of contour values and the number of contour levels. A maximum of 30 contour (NCL) levels is permitted. If CON=OFF, default values are used. In this case, the program will calculate the values for the array and NCL using input data. The default is OFF.

If program set: CALL CONOP3('CON=OFF',0.,0)
If user set: CALL CONOP3('CON=ON',ARRAY,NCL)

Note: The array (ARRAY) contains the contour levels (floating point only) and NCL is the number of levels. The maximum number of contour levels allowed is 30. When assigning the array of contour values, the values must be ordered from smallest to largest.

Example: DATA RLIST(1),...,RLIST(5)/1.,2.,3.,10.,12./

CALL CONOP3('CON=ON',RLIST,5) where 'RLIST' contains the user specified contour levels, and 5 is the number of user specified contour levels (NCL).

Warning on contour options:
It is illegal to use the CON option when either CIL or CHL are activated. If this is done, the option call that detected the error will not be executed.
DAS  This flag determines which contours are represented by dashed lines. The user sets the dashed line pattern. The user may specify that dashed lines be used for contours whose value is less than, equal to, or greater than the dash pattern breakpoint (see the DBP option below), which is zero by default. If DAS=OFF (the default value), all solid lines are used.

All solid lines:  CALL CONOP4('DAS=OFF',' ',0,0)

If greater:  CALL CONOP4('DAS=GTR',PAT,0,0)

If equal:  CALL CONOP4('DAS=EQU',PAT,0,0)

If less:  CALL CONOP4('DAS=LSS',PAT,0,0)

If all same:  CALL CONOP4('DAS=ALL',PAT,0,0)

Note: PAT must be a ten character string with a dollar sign ($) for solid and a single quote ('') for blank. Recall that in FORTRAN 77, in a quoted string a single quote is represented by two single quotes (" ").

Example: CALL CONOP4('DAS=GTR','$$$$$''$$$$$',0,0)

DBP  This flag determines how the dash pattern break point (BP) is set. If DBP=ON, BP must be set by the user by specifying BP. If DBP=OFF the program will set BP to the default value which is zero.

If program set:  CALL CONOP3('DBP=OFF',0.,0)

If user set:  CALL CONOP3('DBP=ON',BP,1)

Note: BP is a floating point number where the break for GTR and LSS contour dash patterns are defined. BP is assumed to be given relative to the untransformed contours.

Example: CALL CONOP3('DBP=ON',5.,1)

where 5. is the user specified break point.

DEF  Reset flags to default values. Activating this option sets all flags to the default value. DEF has no 'ON' or 'OFF' states.

To activate:  CALL CONOP1('DEF')

EXT  Flag to set extrapolation. Normally all CONRAN versions will only plot the boundaries of the convex hull defined by the user's data. To have the contours fill the rectangular area of the frame, set the EXT switch ON. The default is OFF.

To turn on:  CALL CONOP1('EXT=ON')

To turn off:  CALL CONOP1('EXT=OFF')
FMT  Flag for the format of the plotted input data values. If FMT=OFF, the default values for FT, L, and IF are used. The default values are:

FT = '(G10.3)'
L = 7 characters including the parentheses
IF = 10 characters printed in the output field by the format

If FMT=ON, the user must specify values for FT, L, and IF. All user specified values must be given in the correct format.

If program set: CALL CONOP4('FMT=OFF',',',0,0)
If user set: CALL CONOP4('FMT=ON',FT,L,IF)

Note: FT is a character string containing the format. The format must be enclosed in parentheses. Any format, up to 10 characters that is allowed at your installation will be accepted. L is the number of characters in FT. IF is the length of the field created by the format.

Example: CALL CONOP4('FMT=ON','(G30.2)',7,30)

Warning: CONRAS will not test for a valid format. The format is only allowed to be 10 characters long.

GRI  Flag to display the grid. GRI is OFF by default.

To turn on: CALL CONOP1('GRI=ON')
To turn off: CALL CONOP1('GRI=OFF')

Note: If GRI is ON, the virtual grid will be superimposed over the contour plot. The X and Y tick intervals will be displayed under the map only if PER=ON (see PER).

INT  Flag to determine the intensities of the contour lines and other parts of the plot. If INT=OFF, all intensities are set to the default values. If INT=ALL, all intensities are set to the given value, IVAL. If INT is set to one of the other possible options (MAJ, MIN, LAB, or DAT), the intensity level for that option is set to the given value, IVAL.

If program set: CALL CONOP2('INT=OFF',0)
All the same: CALL CONOP2('INT=ALL',IVAL)
Major lines: CALL CONOP2('INT=MAJ',IVAL)
Minor lines: CALL CONOP2('INT=MIN',IVAL)
Title and message: CALL CONOP2('INT=LAB',IVAL)
Data values: CALL CONOP2('INT=DAT',IVAL)

Note: 'INT=DAT' relates to the plotted data values and the plotted maximums and minimums.

Note: IVAL is the intensity desired. IVAL values range from 0 to 255 or the character strings 'LO' and 'HI'. The default is 'HI' except for INT=MIN, which is set to 'LO'.

Example: CALL CONOP2('INT=ALL',110)

**ITP**
Set the interpolation scheme. There are two schemes — C1 surfaces and linear. The C1 method takes longer but will give the best results when the data is sparse (less than 100 points). The linear method will produce a better plot when there is a dense data set. The default is C1 surface.

For C1 surface CALL CONOP1('ITP=C1')

For linear CALL CONOP1('ITP=LIN')

**LAB**
This flag can be set to either label the contours (LAB=ON) or not (LAB=OFF). The default value is LAB=ON.

To turn on: CALL CONOP1('LAB=ON')

To turn off: CALL CONOP1('LAB=OFF')

**LOT**
Flag to list options on the printer. The default value is set to OFF, and no options will be displayed.

To turn on: CALL CONOP1('LOT=ON')

To turn off: CALL CONOP1('LOT=OFF')

Note: If users want to print the option values, they should turn this option ON. The option values will be sent to the standard output unit as defined by the support routine I1MACH.

**LSZ**
This flag determines the label size. If LSZ=OFF, the default ISZLSZ value will be used. If LSZ=ON, the user should specify ISZLSZ. The default value is 9 plotter address units.

If program set: CALL CONOP2('LSZ=OFF',0)

If user set: CALL CONOP2('LSZ=ON',ISZLSZ)
Note: ISZLSZ is the requested character size in plotter address units.

Example: CALL CONOP2('LSZ=ON',4) where 4 is the user desired integer plotter address units.

**MES**  
Flag to plot a message. The default is ON.

To turn on: CALL CONOP1('MES=ON')

To turn off: CALL CONOP1('MES=OFF')

Note: If MES=ON, a message is printed below the plot giving contour intervals and execution time in seconds. If PER or GRI is ON, the message also contains the X and Y tick intervals.

**NCP**  
Flag to indicate the number of data points used for the partial derivative estimation. If NCP=OFF, NUM is set to 4, which is the default value. If NCP=ON, the user must specify NUM greater than or equal to 2.

If program set: CALL CONOP2('NCP=OFF',0)

If user set: CALL CONOP2('NCP=ON',NUM)

Note: NUM = number of data points used for estimation. Changing this value effects the contours produced and the size of input array IWK.

Example: CALL CONOP2('NCP=ON',3)

**PDV**  
Flag to plot the input data values. The default value is PDV=OFF.

To turn on: CALL CONOP1('PDV=ON')

To turn off: CALL CONOP1('PDV=OFF')

Note: If PDV=ON, the input data values are plotted relative to their location on the contour map. If you only wish to see the locations and not the values, set PDV=ON and change FMT to produce an asterisk (*) such as (II).

**PER**  
Flag to set the perimeter. The default value is PER=ON, which causes a perimeter to be drawn around the contour plot.

To turn on: CALL CONOP1('PER=ON')

To turn off: CALL CONOP1('PER=OFF')

Note: If MES is ON, the X and Y tick intervals will be given. These are the intervals in user coordinates that each tick mark represents.
PMM Flag to plot relative minimums and maximums. This flag is OFF by default.
To turn off: CALL CONOP1('PMM=OFF')
To turn on: CALL CONOP1('PMM=ON')

PSL Flag which sets the plot shield option. The outline of the shield will be drawn on the same frame as the contour plot. By default this option is OFF (see SLD option).
Draw the shield: CALL CONOP1('PSL=ON')
Don't draw it: CALL CONOP1('PSL=OFF')

REP Flag indicating the use of the same data in a new execution. The default value is OFF.
To turn on: CALL CONOP1('REP=ON')
To turn off: CALL CONOP1('REP=OFF')
Note: If REP=ON, the same X-Y data and triangulation are to be used but it is assumed the user has changed contour values or resolution for this run. Scratch arrays WK and IWK must remain unchanged.

SCA Flag for scaling of the plot on a frame. This flag is ON by default.
User scaling: CALL CONOP1('SCA=OFF')
Program scaling: CALL CONOP1('SCA=ON')
Prior window: CALL CONOP1('SCA=PRI')
Note: With SCA=OFF, plotting instructions will be issued using the user's input coordinates, unless they are transformed via FX and FY transformations. Users will find an extended discussion in the "INTERFACING WITH OTHER GRAPHICS ROUTINES" section below. The SCA option assumes that all input data falls into the current window setting. With SCA=ON, the entry point will establish a viewport so that the user's plot will fit into the center 90 percent of the frame. When SCA=PRI, the program maps the user's plot instructions into the portion of the frame defined by the current normalization transformation. SCA=OFF should be used to interface with EZMAP.

SDC Flag to determine how to scale the data on the contours. If SDC=OFF, the floating point value is given by scale. If SDC=ON, the user may specify SCALE. The default value for
SCALE is 1.

If program set: CALL CONOP3('SDC=OFF',0.,0)

If user set: CALL CONOP3('SDC=ON',SCALE,1)

Note: The data plotted on contour lines and the data plotted for relative minimums and maximums will be scaled by the floating point value given by SCALE. Typical SCALE values are 10., 100., 1000., etc. The original data values are multiplied by SCALE. SCALE must be a floating point number and is displayed in the message (see MES).

Example: CALL CONOP2('SDC=ON',100.,1)

SLD

Activate or deactivate the shielding option. When this option is activated, only those contours within the shield are drawn. The shield is a polygon specified by the user and must be given in the same coordinate range as the data. It must define only one polygon.

To activate the shield:
CALL CONOP3('SLD=ON',ARRAY,ICSD)

To deactivate the shield:
CALL CONOP3('SLD=OFF',0.,0)

Note: ARRAY is a real array ICSD elements long. The first ICSD/2 elements are X coordinates and the second ICSD/2 elements are Y coordinates. ICSD is the length of entire array, the number of (X + Y) shield coordinates. The polygon must be closed; that is, the first and last points describing it must be the same.

Example: DIMENSION SHLD
DATA SHLD/ 7.,10.,10.,7.,7.,7.,10.,10.,7./
CALL CONOP3('SLD=ON',SHLD,10)

SML

Flag to determine the size of minimum and maximum contour labels. If SML=OFF, the ISZSML default value of 15 is used. If SML=ON, the user must specify ISZSML.

If program set: CALL CONOP2('SML=OFF',0)

If user set: CALL CONOP2('SML=ON',ISZSML)

Note: ISZSML is an integer number that is the size of labels in plotter address units as defined in the SPPS entry WTSTR.

Example: CALL CONOP2('SML=ON',12)
**SPD**

Flag for the size of the plotted input data values. If SPD=OFF, the value of ISZSPD is 8, which is the default. If SPD=ON, the user must specify ISZSPD.

If program set: CALL CONOP2('SPD=OFF',0)

If user set: CALL CONOP2('SPD=ON',ISZSPD)

Note: ISZSPD is an integer number giving the size to plot the data values in plotter address units as defined in the SPPS entry WTSTR.

Example: CALL CONOP2('SPD=ON',6)

**SSZ**

Flag to determine the resolution (number of steps in each direction). If SSZ=ON, the user sets ISTEP, or, if SSZ=OFF, the program will automatically set ISTEP at the default value of 40.

If program set: CALL CONOP2('SSZ=OFF',0)

If user set: CALL CONOP2('SSZ=ON',ISTEP)

Note: ISTEP is an integer specifying the density of the virtual grid. In most cases, the default value of 40 produces pleasing contours. For coarser but quicker contours, lower the value. For smoother contours at the expense of taking longer time, raise the value.

Note: For step sizes greater than 200 in CONRAS, the arrays PV in common CONRA1 and ITLOC in common CONRA9 must be expanded to about 10 more than ISTEP. See CONRA1 and CONRA9 comments below for more information.

Example: CALL CONOP2('SSZ=ON',25)

This ISTEP value will produce a coarse contour.

**STL**

Flag to determine the size of the title. ISZSTL may be set by the user (STL=ON), or the program will set it to the default size of 16 plotter address units (STL=OFF).

If program set: CALL CONOP2('STL=OFF',0)

If user set: CALL CONOP2('STL=ON',ISZSTL)

Note: When 30 or 40 characters are used for the title, the default size of 16 plotter address units works well. For longer titles, a smaller title size is required.

Example: CALL CONOP2('STL=ON',13)
**TEN** Flag to determine the tension factor applied when smoothing contour lines. The user may set TENS or allow the program to set the value. If user set, TENS must have a value greater than zero and less than or equal to 30. The default value is 2.5.

If program set: CALL CONOP3('TEN=OFF',0.,0)

If user set: CALL CONOP3('TEN=ON',TENS,1)

Smoothing of contour lines is accomplished with splines under tension. To adjust the amount of smoothing applied, adjust the tension factor. Setting TENS very large (i.e. 30.), effectively shuts off smoothing.

Example: CALL CONOP3('TEN=ON',14.,1)

**TFR** Flag to advance the frame before triangulation. The default value is TFR=ON, which means that the contours and the triangles will be plotted on separate frames.

If program set: CALL CONOP1('TFR=ON')

To turn off: CALL CONOP1('TFR=OFF')

Note: Triangles are plotted after the contouring is completed. To see the triangles over the contours, turn this switch off.

**TLE** Flag to place a title at the top of the plot. If TLE=ON, the user must specify CHARS and INUM. CHARS is the character string containing the title. INUM is the number of characters in CHARS. The default value is OFF.

To turn on:
CALL CONOP4('TLE=ON',CHARS,INUM,0)

To turn off:
CALL CONOP4('TLE=OFF',' ',0,0)

Note: If longer than 64-character titles are desired, the character variable ISTRNG found in CONRA7 must be increased appropriately.

Example: CALL CONOP4('TLE=ON','VECTOR REVIEW',13,0)

**TOP** Flag to plot only the triangles.

To turn off: CALL CONOP1('TOP=OFF')

To turn on: CALL CONOP1('TOP=ON')

Note: The user may wish to overlay the triangles on some other...
plot. 'TOP=ON' will allow that. This option, when activated (TOP=ON), will set TRI=ON, and TFR=OFF. If the user wants TFR=ON, it should be set after TOP is set. If the user sets TOP=OFF, it will set TRI=OFF and TFR=ON. If the user wants TRI or TFR different, set them after the TOP call.

**TRI** Flag to plot the triangulation. The default is OFF and therefore the triangles are not drawn.

To turn on: CALL CONOP1('TRI=ON')

To turn off: CALL CONOP1('TRI=OFF')

Note: Plotting the triangles will indicate to the user where good and bad points of interpolation are occurring in the contour map. Equilateral triangles are optimal for interpolation. Quality degrades as triangles approach a long and narrow shape. The convex hull of the triangulation is also a poor point of interpolation.

**Option Default Values**

Below are listed the default values for the various options given above. Unless the user specifies otherwise, these values will be used in execution of the various options.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHL</td>
<td>OFF</td>
</tr>
<tr>
<td>CIL</td>
<td>OFF</td>
</tr>
<tr>
<td>CON</td>
<td>OFF</td>
</tr>
<tr>
<td>DAS</td>
<td>OFF</td>
</tr>
<tr>
<td>DBP</td>
<td>OFF</td>
</tr>
<tr>
<td>EXT</td>
<td>OFF</td>
</tr>
<tr>
<td>FMT</td>
<td>OFF</td>
</tr>
<tr>
<td>GRI</td>
<td>OFF</td>
</tr>
<tr>
<td>ITP</td>
<td>C1</td>
</tr>
<tr>
<td>LAB</td>
<td>ON</td>
</tr>
<tr>
<td>LOT</td>
<td>OFF</td>
</tr>
<tr>
<td>LSZ</td>
<td>OFF</td>
</tr>
<tr>
<td>MES</td>
<td>ON</td>
</tr>
<tr>
<td>NCP</td>
<td>OFF</td>
</tr>
<tr>
<td>PDV</td>
<td>OFF</td>
</tr>
<tr>
<td>PER</td>
<td>ON</td>
</tr>
<tr>
<td>PMM</td>
<td>OFF</td>
</tr>
<tr>
<td>REP</td>
<td>OFF</td>
</tr>
<tr>
<td>SCA</td>
<td>OFF</td>
</tr>
<tr>
<td>SLD</td>
<td>OFF</td>
</tr>
<tr>
<td>SML</td>
<td>OFF</td>
</tr>
<tr>
<td>SPD</td>
<td>OFF</td>
</tr>
<tr>
<td>SSZ</td>
<td>OFF</td>
</tr>
<tr>
<td>STL</td>
<td>OFF</td>
</tr>
<tr>
<td>TEN</td>
<td>OFF</td>
</tr>
<tr>
<td>TOP</td>
<td>OFF</td>
</tr>
<tr>
<td>TFR</td>
<td>OFF</td>
</tr>
</tbody>
</table>

**Default Values for User Specified Parameters**

The option default values given above, if used, will set default values for the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>Up to 30 contour levels allowed. Values are computed by the program, based on input.</td>
</tr>
<tr>
<td>BP</td>
<td>0.</td>
</tr>
<tr>
<td>CINC</td>
<td>Computed by the program based on the range of HI and LO values of the input data.</td>
</tr>
</tbody>
</table>
Parameter | Default
---|---
FLO | Computed by the program based on the lowest unscaled input data.
FT | (G10.3) Parentheses must be included.
HI | Computed by the program based on the highest unscaled input data.
CHARS | No title
IF | 10 characters
INUM | No title
IPAT | '$$$$$$$$$$$' (This is a 10 character string.)
ISZLSZ | 9 plotter address units
ISZSML | 15 plotter address units
ISZSPD | 8 plotter address units
ISZSTL | 16 plotter address units
ISTEP | 40
IVAL | 'HI' for all except minor contour lines, which are 'LO'.
L | 7 characters (including both parentheses)
NCL | Computed by the program based on input data. Up to 30 contour levels are permitted.
NUM | 4 data points
SCALE | 1. (no scaling performed)
TENS | 2.5
ICSD | 0 (no shield)

Options Which Effect the Contours

The shape of the contours may be modified by changing NCP and SSZ. NCP controls the number of data points to be used in the interpolation. Increasing NCP causes more of the surrounding data to influence the point of interpolation. Some datasets cause difficulty when trying to produce meaningful contours (triangles that are long and narrow). By modifying NCP a user can fine-tune a plot. Increasing ISTEP, the density of the virtual grid, will smooth out the contour lines and pick up more detail (new contours will appear as ISTEP increases and old ones will
sometimes break into more distinct units). ISTEP is changed by the SSD option.

Note: If NCP.GT.25, arrays DSQ0 and IPC0 in CONDET must be adjusted accordingly. Also NCPSZ in CONBDN (25 by default), must be increased to NCP. The default value of NCP, which is 4, produces pleasing pictures in most cases. However, fine-tuning of the interpolation can be obtained by increasing the size of NCP, with a corresponding linear increase in work space.

The interpolation method used will also cause different-looking contours. The C1 method is recommended when the data is sparse. It will smooth the data and add trends (false hills and valleys). The linear method is recommended when data is dense (GT 50 to 100) it will not smooth the data or add trends.

Interfacing with Other Graphics Routines

Normally the scaling factor will be set to OFF. In most cases mapping can be performed before calling the CONRAS entry point, thus saving the user from modifying the file. If reasonable results cannot be obtained, the statement functions, FX and FY, will have to be replaced. The routines having these statement functions are:

CONDRW, CONPDV, CONTLK, CONPMS, CONGEN

References


CONRAS Error Messages

<table>
<thead>
<tr>
<th>Error</th>
<th>Routine</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONRAS</td>
<td>INPUT PARAMETER NDP LT NCP</td>
</tr>
<tr>
<td>2</td>
<td>CONRAS</td>
<td>NCP GT MAX SIZE OR LT 2</td>
</tr>
<tr>
<td>3</td>
<td>CONTNG</td>
<td>ALL COLINEAR DATA POINTS</td>
</tr>
<tr>
<td>4</td>
<td>CONTNG</td>
<td>IDENTICAL INPUT DATA POINTS FOUND</td>
</tr>
<tr>
<td>5</td>
<td>CONOP</td>
<td>UNDEFINED OPTION</td>
</tr>
<tr>
<td>6</td>
<td>CONCLS</td>
<td>CONSTANT INPUT FIELD</td>
</tr>
<tr>
<td>7</td>
<td>CONOP</td>
<td>INCORRECT CONOP CALL USED</td>
</tr>
<tr>
<td>8</td>
<td>CONOP</td>
<td>ILLEGAL USE OF CON OPTION WITH CIL OR CHL OPTIONS</td>
</tr>
<tr>
<td>9</td>
<td>CONOP</td>
<td>NUMBER OF CONTOUR LEVELS EXCEEDS 30</td>
</tr>
</tbody>
</table>

August 1987
<table>
<thead>
<tr>
<th>Error</th>
<th>Routine</th>
<th>Message</th>
</tr>
</thead>
</table>
| 10    | CONDRW  | CONTOUR STORAGE EXHAUSTED  
This error is trapped and nullified by CONRAN. It serves to signal the user that a contour level may not be complete. |
| 11    | CONSTP  | ASPECT RATIO OF X AND Y GREATER THAN 5 TO 1. This error may cause a poor quality plot. Usually this can be fixed by multiplying X or Y by a constant factor. If this solution is unacceptable then increasing SSZ to a very large value may help. Note: This can be expensive. |

The errors listed above are defined as recoverable errors should the user wish to use them in that fashion. The documentation on the ERPRT77 package explains how to recover from an error.
CONREC
CONREC

SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

Dimension of Arguments
Z(L,N)

Latest Revision
August 1987

Purpose
CONREC draws a contour map from data stored in a rectangular array, labeling the lines.

Usage
If the following assumptions are met, use

CALL EZNCTR (Z,M,N)

ASSUMPTIONS:
• All of the array is to be contoured.
• Contour levels are picked internally.
• Contouring routine picks scale factors.
• Highs and lows are marked.
• Negative lines are drawn with a dashed line pattern.
• EZNCTR calls FRAME after drawing the contour map.

If these assumptions are not met, use

CALL CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

ARGUMENTS

On Input for EZNCTR
Z  M by N array to be contoured.
M  First dimension of Z.
N  Second dimension of Z.

On Output for EZNCTR
All arguments are unchanged.

On Input for CONREC
Z  The (origin of the) array to be contoured. Z is dimensioned L by N.
L  The first dimension of Z in the calling program.
M  The number of data values to be contoured in the X-direction (the first subscript direction). When plotting an entire array, L = M.
N  The number of data values to be contoured in the Y-direction (the second subscript direction).
**FLO** The value of the lowest contour level. If FLO = HI = 0., a value rounded up from the minimum Z is generated by CONREC.

**HI** The value of the highest contour level. If HI = FLO = 0., a value rounded down from the maximum Z is generated by CONREC.

**FINC**

> 0 Increment between contour levels.

= 0 A value, which produces between 10 and 30 contour levels at nice values, is generated by CONREC.

< 0 The number of levels generated by CONREC is ABS(FINC).

**NSET** Flag to control scaling.

= 0 CONREC automatically sets the window and viewport to properly scale the frame to the standard configuration. The GRIDAL entry PERIM is called and tick marks are placed corresponding to the data points.

> 0 CONREC assumes that the user has set the window and viewport in such a way as to properly scale the plotting instructions generated by CONREC. PERIM is not called.

< 0 CONREC generates coordinates so as to place the (untransformed) contour plot within the limits of the user's current window and viewport. PERIM is not called.

**NHI** Flag to control extra information on the contour plot.

= 0 Highs and lows are marked with an H or L as appropriate, and the value of the high or low is plotted under the symbol.

> 0 The data values are plotted at each Z point, with the center of the string indicating the data point location.

< 0 Neither of the above.

**NDOT** A 10-bit constant designating the desired dashed line pattern. If ABS(NDOT) = 0, 1, or 1023, solid lines are drawn. > 0 NDOT pattern is used for all lines. < 0 ABS(NDOT) pattern is used for negative-valued contour lines, and solid is used for positive-valued contours. CONREC converts NDOT to a 16-bit pattern and DASHDB is used. See DASHDB comments in the DASHLINE documentation for details.

**On Output for CONREC** All arguments are unchanged.

**Entry Points** CONREC, CLGEN, REORD, STLINE, DRLINE, MINMAX, PNTVAL, CALCNT, EZNCTR, CONBD
Common Blocks
INTPR, RECINT, CONRE1, CONRE2, CONRE3, CONRE4, CONRE5

Required Library
Standard version: DASHCHAR, which is loaded by default at NCAR.
Smooth version: DASHSMTH, which must be requested at NCAR. Both
versions require GRIDAL, the ERPRT77 package, and the SPPS.

Required GKS Level
0A

I/O Plots contour map.

Precision Single

Language FORTRAN 77

History Replaces old contouring package called CALCNT at NCAR.

Algorithm Each line is followed to completion. Points along a line are found on
boundaries of the (rectangular) cells. These points are connected by line
segments using the software dashed line package, DASHCHAR.
DASHCHAR is also used to label the lines.

Note To draw non-uniform contour levels, see the comments in CLGEN. To
make special modifications for specific needs, see the explanation of the
internal parameters below.

Timing Varies widely with size and smoothness of Z.

Internal Parameters
Name Default Function
ISIZEL 1 Size of line labels, as per the size definitions
given in the SPPS documentation for WTSTR.
ISIZEM 2 Size of labels for minimums and maximums, as
per the size definitions given in the SPPS docu-
mation for WTSTR.
ISIZEP 0 Size of labels for data point values as per the
size definitions given in the SPPS documen-
tation for WTSTR.
NLA 16 Approximate number of contour levels when
internally generated.
NLM 40 Maximum number of contour levels. If this is
to be increased, the dimensions of CL and
RWORK in CONREC must be increased by the
same amount.

August 1987
<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLT</td>
<td>.05</td>
<td>Left hand edge of the plot (0.0 is the left edge of the frame and 1.0 is the right edge of the frame.)</td>
</tr>
<tr>
<td>YBT</td>
<td>.05</td>
<td>Bottom edge of the plot (0.0 is the bottom of the frame and 1.0 is the top of the frame.)</td>
</tr>
<tr>
<td>SIDE</td>
<td>0.9</td>
<td>Length of longer edge of plot (see also EXT).</td>
</tr>
<tr>
<td>NREP</td>
<td>6</td>
<td>Number of repetitions of the dash pattern between line labels.</td>
</tr>
<tr>
<td>NCRT</td>
<td>4</td>
<td>Number of CRT units per element (bit) in the dash pattern.</td>
</tr>
<tr>
<td>ILAB</td>
<td>1</td>
<td>Flag to control the drawing of line labels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ILAB non-zero means label the lines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ILAB = 0 means do not label the lines.</td>
</tr>
<tr>
<td>NULBLL</td>
<td>3</td>
<td>Number of unlabeled lines between labeled lines. For example, when NULBLL = 3, every fourth level is labeled.</td>
</tr>
<tr>
<td>IOFFD</td>
<td>0</td>
<td>Flag to control normalization of label numbers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IOFFD = 0 means include decimal point when possible (do not normalize unless required).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IOFFD non-zero means normalize all label numbers and output a scale factor in the message below the graph.</td>
</tr>
<tr>
<td>EXT</td>
<td>.25</td>
<td>Lengths of the sides of the plot are proportional to M and N (when CONREC sets the window and viewport). In extreme cases, when MIN(M,N)/MAX(M,N) is less than EXT, CONREC produces a square plot.</td>
</tr>
<tr>
<td>IOFFP</td>
<td>0</td>
<td>Flag to control special value feature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IOFFP = 0 means special value feature not in use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IOFFP non-zero means special value feature in use. (SPVAL is set to the special value.) Contour lines will then be omitted from any cell with any corner equal to the special value.</td>
</tr>
<tr>
<td>Name</td>
<td>Default</td>
<td>Function</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>SPVAL</td>
<td>0</td>
<td>Contains the special value when IOFFP is non-zero.</td>
</tr>
</tbody>
</table>
| IOFFM | 0       | Flag to control the message below the plot.  
  * IOFFM = 0 if the message is to be plotted. 
  * IOFFM non-zero if the message is to be omitted. |
| ISOLID| 1023    | Dash pattern for non-negative contour lines.  |
CONRECQCK

SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

Dimension of Arguments

Z(L,N)

Latest Revision

August 1987

Purpose

CONRECQCK draws a contour map from data stored in a rectangular array. No line labels.

Usage

If the following assumptions are met, use

CALL EZCNTR (Z,M,N)

Assumptions:

• All of the array is to be contoured.
• Contour levels are picked internally.
• Contouring routine picks scale factors.
• Highs and lows are marked.
• Negative lines are drawn with a dashed line pattern.
• EZCNTR calls FRAME after drawing the contour map.

If these assumptions are not met, use

CALL CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

ARGUMENTS

On Input for EZCNTR

Z M by N array to be contoured.
M First dimension of Z.
N Second dimension of Z.

On Output for EZCNTR

All arguments are unchanged.

On Input for CONREC

Z The (origin of the) array to be contoured. Z is dimensioned L by N.
L The first dimension of Z in the calling program.
M The number of data values to be contoured in the X-direction (the first subscript direction). When plotting an entire array, L = M.
**N**  The number of data values to be contoured in the Y-direction (the second subscript direction).

**FLO**  The value of the lowest contour level. If FLO = HI = 0., a value rounded up from the minimum Z is generated by CONREC.

**HI**  The value of the highest contour level. If HI = FLO = 0., a value rounded down from the maximum Z is generated by CONREC.

**FINC**

- > 0  Increment between contour levels.
- = 0  A value, which produces between 10 and 30 contour levels at nice values, is generated by CONREC.
- < 0  The number of levels generated by CONREC is ABS(FINC).

**NSET**  Flag to control scaling.

- = 0  CONREC automatically sets the window and viewport to properly scale the frame to the standard configuration. The GRIDAL entry PERIM is called and tick marks are placed corresponding to the data points.
- > 0  CONREC assumes that the user has set the window and viewport in such a way as to properly scale the plotting instructions generated by CONREC. PERIM is not called.
- < 0  CONREC generates coordinates so as to place the (untransformed) contour plot within the limits of the user's current window and viewport. PERIM is not called.

**NHI**  Flag to control extra information on the contour plot.

- = 0  Highs and lows are marked with an H or L as appropriate, and the value of the high or low is plotted under the symbol.
- > 0  The data values are plotted at each Z point, with the center of the string indicating the data point location.
- < 0  Neither of the above.

**NDOT**  A 10-bit constant designating the desired dashed line pattern. If ABS(NDOT) = 0, 1, or 1023, solid lines are drawn.

- > 0  NDOT pattern is used for all lines.
- < 0  ABS(NDOT) pattern is used for negative-valued contour lines, and solid is used for positive-valued contours. CONREC converts NDOT to a 16-bit pattern and DASHDB is used. See DASHDB comments in the DASHLINE documentation for details.
On Output for CONREC

All arguments are unchanged.

Entry Points

CONREC, CLGEN, QUICK, MAXMIN, PNTVAL, CALCNT, EZCNTR, CONBD

Common Blocks

CONRE1, CONRE4, CONRE5

Required Library Routines

GRIDAL, the ERPRT77 package, and the SPPS. DASHCHAR is also needed.

Required GKS Level

0A

Note for NCAR Users

This routine is NOT part of the default libraries at NCAR. CONRECQCK must be acquired, compiled, and loaded to be used at NCAR.

I/O

Plots contour map.

Precision

Single

Language

FORTRAN 77

History

A faster version of CONREC without line labeling capabilities.

Algorithm

The grid space is divided into L*N cells. Each cell is processed in turn, drawing all contour lines in the particular cell until the entire rectangular space is contoured. (This could result in a relatively long plot time on mechanical plotters but does not affect the plot time on CRT-based plotters.)

Note

To draw non-uniform contour levels, see the comments in CLGEN. To make special modifications for specific needs, see the explanation of the internal parameters below.

Timing

Varies widely with size and smoothness of Z.

Internal Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIZEM</td>
<td>2</td>
<td>Size of labels for minimums and maximums, as per the size definitions given in the SPPS documentation for WTSTR.</td>
</tr>
<tr>
<td>ISIZEP</td>
<td>0</td>
<td>Size of labels for data point values as per the size definitions given in the SPPS documentation for WTSTR.</td>
</tr>
<tr>
<td>NLA</td>
<td>16</td>
<td>Approximate number of contour levels when internally generated.</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>NLM</td>
<td>40</td>
<td>Maximum number of contour levels. If this is to be increased, the dimensions of CL and RWORK in CONREC must be increased by the same amount.</td>
</tr>
<tr>
<td>XLT</td>
<td>.05</td>
<td>Left hand edge of the plot (0.0 is the left edge of the frame and 1.0 is the right edge of the frame.)</td>
</tr>
<tr>
<td>YBT</td>
<td>.05</td>
<td>Bottom edge of the plot (0.0 is the bottom of the frame and 1.0 is the top of the frame.)</td>
</tr>
<tr>
<td>SIDE</td>
<td>0.9</td>
<td>Length of longer edge of plot (see also EXT).</td>
</tr>
</tbody>
</table>
| IOFFD    | 0     | Flag to control normalization of label numbers.  
  - IOFFD = 0 means include decimal point when possible (do not normalize unless required).  
  - IOFFD non-zero means normalize all label numbers and output a scale factor in the message below the graph. |
| EXT      | .25   | Lengths of the sides of the plot are proportional to M and N (when CONREC sets the window and viewport). In extreme cases, when MIN(M,N)/MAX(M,N) is less than EXT, CONREC produces a square plot. |
| IOFFP    | 0     | Flag to control special value feature.  
  - IOFFP = 0 means special value feature not in use.  
  - IOFFP non-zero means special value feature in use. (SPVAL is set to the special value.) Contour lines will then be omitted from any cell with any corner equal to the special value. |
| SPVAL    | 0     | Contains the special value when IOFFP is non-zero. |
| IOFFM    | 0     | Flag to control the message below the plot.  
  - IOFFM = 0 if the message is to be plotted.  
  - IOFFM non-zero if the message is to be omitted. |
CONRECSPR
CONRECSPR

SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

Dimension of Arguments Z(L,N)

Latest Revision August 1987

Purpose CONRECSPR draws a contour map from data stored in a rectangular array, labeling the lines. This is the so-called "super" version, which smooths contour lines and removes crowded lines.

Usage If the following assumptions are met, use

CALL EZCNTR (Z,M,N)

ASSUMPTIONS:
• All of the array is to be contoured.
• Contour levels are picked internally.
• Contouring routine picks scale factors.
• Highs and lows are marked.
• Negative lines are drawn with a dashed line pattern.
• EZCNTR calls FRAME after drawing the contour map.

If these assumptions are not met, use

CALL CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

ARGUMENTS

On Input for EZCNTR
Z M by N array to be contoured.
M First dimension of Z.
N Second dimension of Z.

On Output for EZCNTR
All arguments are unchanged.

On Input for CONREC
Z The (origin of the) array to be contoured. Z is dimensioned L by N.
L The first dimension of Z in the calling program.
M The number of data values to be contoured in the X-direction (the first subscript direction). When plotting an entire array, L = M.
N The number of data values to be contoured in the Y-direction (the second subscript direction).
FLO  The value of the lowest contour level. If FLO = HI = 0., a value
rounded up from the minimum Z is generated by CONREC.

HI   The value of the highest contour level. If HI = FLO = 0., a
value rounded down from the maximum Z is generated by
CONREC.

FINC

> 0  Increment between contour levels.

= 0  A value, which produces between 10 and 30 contour levels
at nice values, is generated by CONREC.

< 0  The number of levels generated by CONREC is
ABS(FINC).

NSET  Flag to control scaling.

= 0  CONREC automatically sets the window and viewport to
properly scale the frame to the standard configuration. The
GRIDAL entry PERIM is called and tick marks are
placed corresponding to the data points.

> 0  CONREC assumes that the user has set the window and
viewport in such a way as to properly scale the plotting in-
structions generated by CONREC. PERIM is not called.

< 0  CONREC generates coordinates so as to place the (un-
transformed) contour plot within the limits of the user's
current window and viewport. PERIM is not called.

NHI   Flag to control extra information on the contour plot.

= 0  Highs and lows are marked with an H or L as appropriate,
and the value of the high or low is plotted under the sym-
bol.

> 0  The data values are plotted at each Z point, with the
center of the string indicating the data point location.

< 0  Neither of the above.

NDOT  A 10-bit constant designating the desired dashed line pattern. If
ABS(NDOT) = 0, 1, or 1023, solid lines are drawn.

> 0  NDOT pattern is used for all lines.

< 0  ABS(NDOT) pattern is used for negative-valued contour
lines, and solid is used for positive-valued contours.
CONREC converts NDOT to a 16-bit pattern and
DASHDB is used. See DASHDB comments in the DASH-
LINE documentation for details.
On Output for CONREC

All arguments are unchanged.

Entry Points

CONREC, CLGEN, REORD, STLINE, DRLINE, MINMAX, PNTVAL, CALCNT, EZCNTR, CONBD

Common Blocks

INTPR, SPRINT, CONRE1, CONRE2, CONRE3, CONRE4, CONRE5

Required Library Routines

GRIDAL, the ERPRT77 package, and the SPPS. DASHSUPR is also needed.

Required GKS Level

0A

Note for NCAR Users

This routine is NOT part of the default libraries at NCAR. CONRECSPR must be acquired, compiled, and loaded to be used at NCAR.

I/O

Plots contour map.

Precision

Single

Language

FORTRAN 77

History

Algorithm

Each line is followed to completion. Points along a line are found on boundaries of the (rectangular) cells. These points are connected by line segments using the software dashed line package, DASHSUPR. DASHSUPR is also used to label the lines. In this version, a model of the plotting plane is maintained. If a line to be drawn will overlap previously drawn lines, it is omitted.

Note

To draw non-uniform contour levels, see the comments in CLGEN. To make special modifications for specific needs, see the explanation of the internal parameters below.

Timing

Varies widely with size and smoothness of Z.

Internal Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIZEL</td>
<td>1</td>
<td>Size of line labels, as per the size definitions given in the SPPS documentation for WTSTR.</td>
</tr>
<tr>
<td>ISIZEM</td>
<td>2</td>
<td>Size of labels for minimums and maximums as per the size definitions given in the SPPS documentation for WTSTR.</td>
</tr>
<tr>
<td>Name</td>
<td>Default</td>
<td>Function</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ISIZEP</td>
<td>0</td>
<td>Size of labels for data point values as per the size definitions given in the SPPS documentation for WTSTR.</td>
</tr>
<tr>
<td>NLA</td>
<td>16</td>
<td>Approximate number of contour levels when internally generated.</td>
</tr>
<tr>
<td>NLM</td>
<td>40</td>
<td>Maximum number of contour levels. If this is to be increased, the dimensions of CL and RWORK in CONREC must be increased by the same amount.</td>
</tr>
<tr>
<td>XLT</td>
<td>.05</td>
<td>Left hand edge of the plot (0.0 is the left edge of the frame and 1.0 is the right edge of the frame.)</td>
</tr>
<tr>
<td>YBT</td>
<td>.05</td>
<td>Bottom edge of the plot (0.0 is the bottom of the frame and 1.0 is the top of the frame.)</td>
</tr>
<tr>
<td>SIDE</td>
<td>0.9</td>
<td>Length of longer edge of plot (see also EXT).</td>
</tr>
<tr>
<td>NREP</td>
<td>6</td>
<td>Number of repetitions of the dash pattern between line labels.</td>
</tr>
<tr>
<td>NCRT</td>
<td>4</td>
<td>Number of CRT units per element (bit) in the dash pattern.</td>
</tr>
<tr>
<td>ILAB</td>
<td>1</td>
<td>Flag to control the drawing of line labels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ILAB non-zero means label the lines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ILAB = 0 means do not label the lines.</td>
</tr>
<tr>
<td>NULBLL</td>
<td>3</td>
<td>Number of unlabeled lines between labeled lines. For example, when NULBLL = 3, every fourth level is labeled.</td>
</tr>
<tr>
<td>IOFFD</td>
<td>0</td>
<td>Flag to control normalization of label numbers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IOFFD = 0 means include decimal point when possible (do not normalize unless required).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IOFFD non-zero means normalize all label numbers and output a scale factor in the message below the graph.</td>
</tr>
<tr>
<td>Name</td>
<td>Default</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>EXT</td>
<td>.25</td>
<td>Lengths of the sides of the plot are proportional to M and N (when CONREC sets the window and viewport). In extreme cases, when MIN(M,N)/MAX(M,N) is less than EXT, CONREC produces a square plot.</td>
</tr>
</tbody>
</table>
| IOFFP| 0       | Flag to control special value feature.  
* IOFFP = 0 means special value feature not in use.  
* IOFFP non-zero means special value feature in use. (SPVAL is set to the special value.) Contour lines will then be omitted from any cell with any corner equal to the special value. |
| SPVAL| 0       | Contains the special value when IOFFP is non-zero. |
| IOFFM| 0       | Flag to control the message below the plot.  
* IOFFM = 0 if the message is to be plotted.  
* IOFFM non-zero if the message is to be omitted. |
| ISOLID| 1023   | Dash pattern for non-negative contour lines. |
| IHIL0| 3       | Flag to control labeling of highs, lows, or both: if NHI = 0, then  
* IHIL0 = 0 means do not label highs nor lows  
* IHIL0 = 1 means highs are labeled, lows are not  
* IHIL0 = 2 means lows are labeled, highs are not  
* IHIL0 = 3 means both highs and lows are labeled |
DASHCHAR

Software dashed-line package with character capability

Latest Revision August 1987

Purpose DASHCHAR is a software dashed-line package.

Usage First, either
CALL DASHDB (IPAT)
where IPAT is a 16-bit dash pattern as described in the subroutine
DASHDB (see DASHLINE documentation in this manual), or
CALL DASHDC (IPAT,JCRT,JSIZE)
as described below.

Then, call any of the following:
CALL CURVED (X,Y,N)
CALL FRSTD (X,Y)
CALL VECTD (X,Y)
CALL LASTD

LASTD is called only after the last point of a line has been processed in
VECTD.

The following may also be called, but no smoothing will result:
CALL LINED (XA,YA,XB,YB)

SUBROUTINE DASHDC (IPAT,JCRT,JSIZE)

ARGUMENTS

On Input to DASHDC

IPAT A character string of arbitrary length (60 characters seems to be a practical limit) which specifies the dash pattern to be used. A dollar sign in IPAT indicates solid; an apostrophe indicates a gap; blanks are ignored. Any character in IPAT which is not a dollar sign, apostrophe, or blank is considered to be part of a line label. Each line label can be at most 15 characters in length. Sufficient white space is reserved in the dashed line for writing line labels.

JCRT The length in plotter address units per $ or apostrophe.

JSIZE Is the size of the plotted characters:
- If between 0 and 3, it is 1., 1.5, 2., and 3. times an 8 plotter address unit width.
- If greater than 3, it is the character width in plotter address units.
To Other Line-Drawing Routines

CURVED(X,Y,N)  X and Y are arrays of world coordinate values of length N or greater. Line segments obeying the specified dash pattern are drawn to connect the N points.

FRSTD(X,Y)  The current pen position is set to the world coordinate value (X,Y).

VECTD(X,Y)  A line segment is drawn between the world coordinate value (X,Y) and the most recent pen position. (X,Y) then becomes the most recent pen position.

LINED(XA,XB,YA,YB)  A line is drawn between world coordinate values (XA,YA) and (XB,YB).

LASTD  When using FRSTD and VECTD, LASTD may be called (no arguments needed). If the dashed line package was leaving a space for characters, and the line ended before there was enough space for the characters, the space left will be filled in if LASTD is called.

On Output  All arguments are unchanged for all routines.

Note  When switching from the regular plotting routines to a dashed-line package the first call should not be to VECTD.

Entry Points  DASHDB, DASHDC, CURVED, FRSTD, VECTD, LINED, RESET, LASTD, CFVLD, DRAWPV, DASHBD

Common Blocks  INTPR, DASHD1, DASHD2, DDFLAG, DCFLAG, DSAVE1, DSAVE2, DSAVE3, CFFLAG, DCFLAG

Required Library Routines  The ERPRT77 package and the SPPS.

Required Level Of GKS  0A

I/O  Plots solid or dashed lines, possibly with characters at intervals in the line.

Precision  Single

Language  FORTRAN 77

Algorithm
The position in the dash pattern is remembered as points are processed. The distance transversed in plotter address space is used to determine whether to draw segments, parts of segments, characters, or nothing.

Accuracy
Plus or minus .5 plotter address units per call. There is no cumulative error.

Timing
For solid or blank lines, there is almost no overhead. Dashed lines take about 4 times as long as drawing solid lines. (The line drawing software is so fast that the increase will not be noticed in most programs.) Patterns with characters do not take much longer than those without.

Portability
FORTRAN 77

<table>
<thead>
<tr>
<th>Internal Parameters</th>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPAU</td>
<td>3</td>
<td>Number of plotter addresses per element in the dash pattern for solid lines and gaps. The pattern is repeated every IPAU*16 plotter address units.</td>
<td></td>
</tr>
<tr>
<td>FPART</td>
<td>1</td>
<td>Multiplicative factor for first solid-line segment. This can be used to off-set labels. For example, if FPART = .5, the first solid line segment is only one-half as long as the other solid-line segments. This moves all labels on this line towards the beginning, which reduces the probability of the label being written on top of a label of a nearby line drawn with FPART = 1.</td>
<td></td>
</tr>
</tbody>
</table>
| IGP                 | 9    | Flag to control whether a gap is left for characters when plotting.  
= 9  Gap is left.  
= 0  No gap is left. |
| ICLOSE              | 6    | An internal or external call to set the pen (pen-up) to a specific position is executed only if this position is more than ICLOSE metacode address units away from the current pen position (distance=difference in X-coordinates + difference in Y-coordinates). |
DASHLINE

Software dashed-line package

Latest Revision August 1987

Purpose DASHLINE is a software dashed-line package. Some hardware dashed line generators fail to produce pleasing results when drawing very short vector segments or vector segments of varying lengths. This package does not have this problem.

Usage First,
CALL DASHDB (IPAT)

Then, call any of the following:
CALL CURVED (X,Y,N)
CALL FRSTD (X,Y)
CALL VECTD (X,Y)
CALL LINED (XA,YA,XB,YB)

SUBROUTINE DASHDB (IPAT)

ARGUMENTS

On Input DASHDB (IPAT) IPAT is a 16-bit dash pattern. By default each bit in the pattern represents 3 plotter address units (1=solid, 0=blank)

To Other Line-Drawing Routines CURVED(X,Y,N) X and Y are arrays of world coordinate values of length N or greater. Line segments obeying the specified dash pattern are drawn to connect the N points.
FRSTD(X,Y) The current pen position is set to the world coordinate value (X,Y)
VECTD(X,Y) A line segment is drawn between the world coordinate value (X,Y) and the most recent pen position. (X,Y) then becomes the most recent pen position.
LINED(XA,XB,YA,YB) A line is drawn between world coordinate values (XA,YA) and (XB,YB).

On Output All arguments are unchanged for all routines.

Entry Points DASHDB, CURVED, FRSTD, VECTD, LINED, LASTD, CFVLD, DRAWPV, DASHBD

August 1987
**Common Blocks**  
INTPR, DSHD, DSHDA, DSHDD, DSHDC, DSHDB

**Required Library**  
The ERPRT77 package and the SPPS.

**Required GKS Level**  
0A

**Precision**  
Single

**Language**  
FORTRAN 77

**History**  

**Algorithm**  
The position in the dash pattern is remembered as points are processed. The distance transversed in plotter address space is used to determine whether to draw segments, parts of segments, characters, or nothing. The plotter address space is 1024 X 1024.

**Accuracy**  
Plus or minus .5 plotter address units per call. There is no cumulative error.

**Timing**  
For solid or blank lines, there is almost no overhead. Dashed lines take about 4 times as long as drawing solid lines. (The line drawing software is so fast that the increase will not be noticed in most programs.)

**Portability**  
FORTRAN 77

**Internal Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPAU</td>
<td>3</td>
<td>Number of plotter addresses per element in the dash pattern for solid lines and gaps. The pattern is repeated every IPAU*16 plotter address units.</td>
</tr>
<tr>
<td>ICLOSE</td>
<td>6</td>
<td>An internal or external call to set the pen (pen-up) to a specific position is executed only if this position is more than ICLOSE metacode address units away from the current pen position (distance= difference in X-coordinates + difference in Y-coordinates).</td>
</tr>
</tbody>
</table>
DASHSMTH

Software dashed-line package with character capability and smoothing

Latest Revision August 1987

Purpose DASHSMTH is a software dashed-line package with smoothing capabilities. DASHSMTH is DASHCHAR with smoothing features added.

Usage First, either
CALL DASHDB (IPAT)
where IPAT is a 16-bit dash pattern as described in the subroutine
DASHDB (see DASHLINE documentation), or
CALL DASHDC (IPAT,JCRT,JSIZE)
as described below.

Then, call any of the following:
CALL CURVED (X,Y,N)
CALL FRSTD (X,Y)
CALL VECTD (X,Y)
CALL LASTD
LASTD is called only after the last point of a line has been processed in VECTD.

The following may also be called, but no smoothing will result:
CALL LINED (XA,YA,XB,YB)

ARGUMENTS

On Input to
DASHDC

IPAT A character string of arbitrary length (60 characters seems to be a practical limit) that specifies the dash pattern to be used. A dollar sign in IPAT indicates solid; an apostrophe indicates a gap; blanks are ignored. Any character in IPAT that is not a dollar sign, apostrophe, or blank is considered to be part of a line label. Each line label can be at most 15 characters in length. Sufficient white space is reserved in the dashed line for writing line labels.

JCRT The length in plotter address units per $ or apostrophe.

JSIZE Is the size of the plotted characters:
  • If between 0 and 3, it is 1., 1.5, 2. and 3. times an 8 plotter address unit width.
  • If greater than 3, it is the character width in plotter address units.
### To Other Line-Drawing Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURVED(X,Y,N)</td>
<td>X and Y are arrays of world coordinate values of length N or greater. Line segments obeying the specified dash pattern are drawn to connect the N points.</td>
</tr>
<tr>
<td>FRSTD(X,Y)</td>
<td>The current pen position is set to the world coordinate value (X,Y).</td>
</tr>
<tr>
<td>VECTD(X,Y)</td>
<td>A line segment is drawn between the world coordinate value (X,Y) and the most recent pen position. (X,Y) then becomes the most recent pen position.</td>
</tr>
<tr>
<td>LINED(XA,XB,YA,YB)</td>
<td>A line is drawn between world coordinate values (XA,YA) and (XB,YB).</td>
</tr>
<tr>
<td>LASTD</td>
<td>When using FRSTD and VECTD, LASTD must be called (no arguments needed). LASTD sets up the calls to the smoothing routines KURV1S and KURV2S.</td>
</tr>
</tbody>
</table>

### On Output

All arguments are unchanged for all routines.

### Note

When switching from the regular plotting routines to a dashed-line package the first call should not be to VECTD.

### Entry Points

DASHDB, DASHDC, CURVED, FRSTD, VECTD, LINED, RESET, LASTD, KURV1S, KURV2S, CFVLD, FDVDLD, DRAWPV, DASHBD

### Common Blocks

INTPR, DASH1, DASH2, DDFLAG, DCFLAG, DSAVE1, DSAVE2, DSAVE3, DSAVE5, CFFLAG, SMFLAG, DFFLAG, FDFLAG

### Required Library Routines

The ERPRT77 package and the SPPS.

### Required GKS Level

0A

### I/O

Plots solid or dashed lines, possibly with characters at intervals in the line. The lines may also be smoothed.

### Precision

Single

### Language

FORTRAN 77

### History

Algorithm Points for each line segment are processed and passed to the routines, KURV1S and KURV2S, which compute splines under tension passing through these points. New points are generated between the given points, resulting in smooth lines.

Accuracy Plus or minus .5 plotter address units per call. There is no cumulative error.

Timing About three times as long as DASHCHAR.
DASHSUPR
DASHSUPR

Software dashed-line package with character capability, smoothing, and the capability of removing crowded lines.

Latest Revision August 1987

Purpose DASHSUPR is a software dashed-line package with smoothing capabilities and the capability of removing crowded lines.

Usage First, do
CALL RESET

Then, either
CALL DASHDB (IPAT)
where IPAT is a 16-bit dash pattern as described in the subroutine DASHDB (see DASHLINE documentation), or
CALL DASHDC (IPAT,JCRT,JSIZE)
as described below.

Then, call any of the following:
CALL CURVED (X,Y,N)
CALL FRSTD (X,Y)
CALL VECTD (X,Y)
CALL LASTD

LASTD is called only after the last point of a line has been processed in VECTD.

The following may also be called, but no smoothing will result:
CALL LINED (XA,YA,XB,YB)

PWRTM can be called to draw characters and mark, in the model picture, the regions where the characters have been drawn.
CALL PWRTM (X,Y,IDPC,ISIZ,IOR,ICNT)

For details, see the subroutine PWRTM in this package.

ARGUMENTS

On Input to DASHDC

IPAT A character string of arbitrary length (60 characters seems to be a practical limit) that specifies the dash pattern to be used. A dollar sign in IPAT indicates solid; an apostrophe indicates a gap; blanks are ignored. Any character in IPAT that is not a dollar sign, apostrophe, or blank is considered to be part of a line label. Each line label can be at most 15 characters in length. Sufficient white space is reserved in the dashed line for writing line labels.

JCRT The length in plotter address units per $ or apostrophe.

JSIZE Is the size of the plotted characters:
To Other Line-Drawing Routines

CURVED(X,Y,N) X and Y are arrays of world coordinate values of length N or greater. Line segments obeying the specified dash pattern are drawn to connect the N points.

FRSTD(X,Y) The current pen position is set to the world coordinate value (X,Y).

VECTD(X,Y) A line segment is drawn between the world coordinate value (X,Y) and the most recent pen position. (X,Y) then becomes the most recent pen position.

LINED(XA,XB,YA,YB) A line is drawn between world coordinate values (XA,YA) and (XB,YB).

LASTD When using FRSTD and VECTD, LASTD must be called (no arguments needed). LASTD sets up the calls to the smoothing routines KURVIS and KURV2S.

PWRTM (X,Y,IDPC,ISIZ,IOR,ICNT) The arguments for PWRTM are the same as those for the utility support routine WTSTR.

On Output

All arguments are unchanged for all routines.

Note

When switching from the regular plotting routines to a dashed-line package the first call should not be to VECTD.

Entry Points

DASHDB, DASHDC, CURVED, FRSTD, VECTD, LINED, RESET, LASTD, KURVIS, KURV2S, CFVD, FDVDLD, DRAWPV, DASHBD, CUTUP, REMOVE, MARKL, PWRTM.

Common Blocks

INTPR, DASHD1, DASHD2, DDFLAG, DCFLAG, DSAVE1, DSAVE2, DSAVE3, DSAVE4, DSAVE5, CFFLAG, SMFLAG, DFFLAG.

Required Library Routines

The ERPRT77 package and the SPPS.

Required GKS Level

0A

I/O

Plots solid or dashed lines, possibly with characters at intervals in the line. The lines may also be smoothed.

Precision

Single
Language FORTRAN 77

History Originally written by Tom Reid at Texas A & M. Made portable in 1977 for all machines that support plotters with up to 15 bit resolution. Converted to FORTRAN 77 and GKS in June 1984.

Algorithm Points for each line segment are processed and passed to the routines, KURV1S and KURV2S, which compute splines under tension passing through these points. New points are generated between the given points, resulting in smooth lines. As each line is drawn, a test is done to see if that part of the plotting plane has been drawn on. If it has, then that line, or part of that line, is not drawn. As the lines are drawn, they are also marked into the model picture.

The model picture is a bit map, 1024 X 1024, which marks each pixel as it is used.

Note: The user should draw all lines of major importance first and then lines of minor importance since the algorithm removes the last line drawn if line removal is necessary.

Accuracy Plus or minus .5 plotter address units per call. There is no cumulative error.

Timing About four times as long as DASHCHAR.

Portability FORTRAN 77

<table>
<thead>
<tr>
<th>Internal Parameters</th>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IPAU</td>
<td>3</td>
<td>Number of plotter addresses per element in the dash pattern for solid lines and gaps. The pattern is repeated every IPAU*16 plotter address units.</td>
</tr>
<tr>
<td></td>
<td>FPART</td>
<td>1</td>
<td>Multiplicative factor for first solid-line segment. This can be used to off-set labels. For example, if FPART = .5, the first solid line segment is only one-half as long as the other solid-line segments. This moves all labels on this line towards the beginning, which reduces the probability of the label being written on top of a label of a nearby line drawn with FPART = 1.</td>
</tr>
</tbody>
</table>
|                     | IGP | 9       | Flag to control whether a gap is left for characters when plotting.  
= 9 Gap is left.  
= 0 No gap is left. |
<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
</table>
| IOFFS  | 0       | Flag to turn on smoothing code.  
          | = 0  Smoothing.  
<pre><code>      | = 1  No smoothing.                                                                               |
</code></pre>
<p>| TENSN  | 2.5     | Tension factor. Must be greater than 0. A large tension factor (30.) would essentially turn off smoothing. |
| NP     | 150     | Twice the maximum number of interpolated points on a horizontal line with length equal to that of the grid. More points per unit length are interpolated for short lines than for long lines. |
| SMALL  | 128.    | Minimum distance in metacode address units between points that are saved. When the points on a line are being processed, only the first of two consecutive points is saved if the points are less than SMALL metacode address units apart. This procedure is to prevent cusps. |
| L1     | 70      | The maximum number of points saved at one time. If there are more than L1 points on a given line, L1 points are processed, then the next L1, until the entire line is processed. Smoothness between segments is maintained automatically. If L1 is increased, the dimensions of XSAVE, YSAVE, XP, YP, and TEMP in FDVDLD must be increased to the new value of L1. |
| ADDLR  | 2.      | Number of plotter address units added to each character string, on the left and the right, as free space. |
| ADDTB  | 2.      | Number of plotter address units added to each character string, on the top and on the bottom, as free space. |
| MLLINE | 384     | The maximum length in each coordinate of a single line to be processed by the crowded line algorithm. Lines longer than MLLINE metacode address units in a coordinate are broken into smaller segments, each of which is processed separately. This is done to prevent anomalies in the removal of long lines since only the starting point and end point of each line is checked in that process. |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLOSE</td>
<td>6</td>
<td>An internal or external call to set the pen (pen-up) to a specific position is executed only if this position is more than ICLOSE metacode address units away from the current pen position (distance=difference in X-coordinates + difference in Y-coordinates).</td>
</tr>
</tbody>
</table>
EZMAP
EZMAP

Introduction .................................................................................................. 205
Projections .................................................................................................... 206
  Conical Projections .................................................................................. 207
  Azimuthal Projections ............................................................................. 208
  Cylindrical Projections ......................................................................... 209
Error Conditions ............................................................................................ 211
Implementation .............................................................................................. 212
History ........................................................................................................... 212
I/O ................................................................................................................... 212
Accuracy .......................................................................................................... 213
Timing ............................................................................................................. 213
References ........................................................................................................ 213
Routines .......................................................................................................... 213
  MAPDRW ................................................................................................... 213
  MAPEOD (NOUT,NSEG,IDLS,IDRS,NPTS,PNTS) ....................................... 214
  MAPFST (RLAT,RLON) ........................................................................... 214
  MAPGRD ................................................................................................... 215
  MAPGTx (WHCH,xVAL) ........................................................................... 215
  MAPINT ..................................................................................................... 217
  MAPIO ....................................................................................................... 217
  MAPIT (RLAT,RLON,IFST) .................................................................... 217
  MAPLBI ................................................................................................... 218
  MAPLOT .................................................................................................... 218
  MAPPOS (XLOW,XROW,YBOW,YTOW) ................................................ 219
  MAPROJ (JPRJ,PLAT,PLON,ROTA) ................................................... 219
  MAPRS .................................................................................................... 221
  MAPRST (IFNO) ....................................................................................... 221
  MAPSAV (IFNO) ...................................................................................... 222
  MAPSET (JLTS,PLM1,PLM2,PLM3,PLM4) ........................................... 222
  MAPSTx (WHCH,xVAL) ......................................................................... 222
  MAPTRN (RLAT,RLON,UVAL,VVAL) ................................................ 226
  MAPUSR (IPRT) ....................................................................................... 227
  MAPVEC (RLAT,RLON) ......................................................................... 227
  SUPCON (RLAT,RLON,UVAL,VVAL) ................................................ 228
  SUPMAP (...) ......................................................................................... 228
<table>
<thead>
<tr>
<th>Examples</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1</td>
<td>232</td>
</tr>
<tr>
<td>Example 2</td>
<td>234</td>
</tr>
<tr>
<td>Example 3</td>
<td>236</td>
</tr>
<tr>
<td>Example 4</td>
<td>239</td>
</tr>
<tr>
<td>Example 5</td>
<td>242</td>
</tr>
<tr>
<td>Example 6</td>
<td>246</td>
</tr>
<tr>
<td>Example 7</td>
<td>249</td>
</tr>
<tr>
<td>Example 8</td>
<td>255</td>
</tr>
<tr>
<td>Example 9</td>
<td>258</td>
</tr>
<tr>
<td>A Final Example</td>
<td>268</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area Identifiers</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>271</td>
</tr>
</tbody>
</table>
SUBROUTINE EZMAP

INTRODUCTION
The EZMAP package allows one to plot maps of the earth according to any of ten different projections, with parallels, meridians, and continental, international, and/or U.S. state outlines. The origin and orientation of the projection are selected by the user. Points on the earth defined by latitude and longitude are mapped to points in the plane of projection - the u/v plane. A rectangular frame whose sides are parallel to the u and v axes is chosen; material within that frame (or an inscribed elliptical frame) is mapped to the plane of the plotter - the x/y plane - for plotting. The u and v axes are parallel to the x and y axes, respectively.

Two changes were made in EZMAP in July 1987: MAPEOS was replaced by MAPEOD and the seven intensity parameters were replaced by color index parameters. Also, the EZMAP example codes were rewritten for execution in a GKS environment.

To draw a complete map, as directed by the current values of the EZMAP parameters, the user need only execute the statement "CALL MAPDRW." Thus, the principal EZMAP routine is this:

MAPDRW Draw a complete map. MAPDRW calls four routines. In some situations, the user may wish to call them directly. These routines are as follows:

MAPINT Initialize. MAPINT must be called initially and again after one or more of MAPPOS, MAPROJ, and MAPSET has been called.

MAPGRD Draws selected parallels and meridians.

MAPLBL Labels the international date line, the equator, the Greenwich meridian, and the poles, and draws the perimeter.

MAPLOT Draws selected geographic outlines.

The following routines are used to change the values of EZMAP parameters, and thus the appearance of the map:

MAPPOS Determine what portion of the plotter frame is to be used.

MAPROJ Determine the projection to be used.

MAPSET Determine what portion of the u/v plane is to be viewed.

MAPSTC Set a parameter of type CHARACTER.

MAPSTI Set a parameter of type INTEGER.

MAPSTL Set a parameter of type LOGICAL.

MAPSTR Set a parameter of type REAL.

The following routines are used to retrieve the current values of EZMAP parameters:

MAPGTC Get a parameter of type CHARACTER.

MAPGTI Get a parameter of type INTEGER.
MAPGTL  Get a parameter of type LOGICAL.
MAPGTR  Get a parameter of type REAL.

To save/restore the current values of EZMAP parameters, use the following:
MAPSAV  To save the values.
MAPRST  To restore saved values.

To draw objects on the map, use the following routines:
MAPTRN  To compute the u/v coordinates of a point from its latitude and longitude.
MAPIT    To do "pen-up" or "pen-down" moves.
MAPFST   To do "pen-up" moves.
MAPVEC   To do "pen-down" moves.
MAPIQ    To signal the end of a string of calls to MAPIT and to cause its buffers to be flushed.

To re-execute the "CALL SET" originally executed by MAPINT, use this routine:
MAPRS    To re-execute the "CALL SET".

By default, grid lines are drawn using software dashed lines and geographical outlines are drawn using either solid lines or dotted lines. The dash pattern used for the grid lines, the flag which says whether outlines are solid or dotted, and the color indices of various parts of the map are all user-settable parameters. More complete control of color indices, spot size, dash pattern, etc., may be achieved by supplying one's own version of the routine MAPUSR, which is called by EZMAP just before and just after drawing various parts of the map; a user version of this routine may be as complicated as is required to achieve a desired effect.

The routine MAPEOD is called by EZMAP once for each segment in the outline dataset. The user may supply a version which examines the segment to see if it ought to be plotted and, if not, to delete it. This may be used (for example) to reduce the clutter in northern Canada.

The routine SUPMAP, from which EZMAP grew, is implemented within it and allows one to draw a complete map with a single, rather lengthy, call. The routine SUPCON, which is the old analogue of MAPTRN, is also implemented.

PROJECTIONS  EZMAP offers ten different projections, in three groups: conical, azimuthal, and cylindrical.
Conical Projections

Conical projections map the surface of the earth onto a cone which is either tangent to the earth along a single circle or intersects it along two circles. The cone is cut along some line passing through its vertex and opened up onto a flat surface. The only conical projection offered by EZMAP is the Lambert conformal with two standard parallels (becoming a simple conic with one standard parallel if the two parallels are made equal). The cone intersects the earth along two user-specified standard parallels (lines of latitude), which would normally both be in the Northern Hemisphere or in the Southern Hemisphere; the cone is cut along the line opposite a user-specified central meridian (line of longitude) and laid flat on the u/v plane with either the North Pole or the South Pole (as implied by the standard parallels) at the origin.

If LAT1 and LAT2 are the latitudes of the two standard parallels and LAT1 is not equal to LAT2, the so-called "cone constant" is given by the formula

$$\text{CONE} = \frac{\log(\cos(LAT1)) - \log(\cos(LAT2))}{\log(\tan(45-S*LAT1/2)) - \log(\tan(45-S*LAT2/2))}$$

where "S" is +1 in the Northern Hemisphere and -1 in the Southern Hemisphere. If LAT1 equals LAT2, then

$$\text{CONE} = \cos(90-S*LAT1)$$

The value of CONE is between 0 and 1; CONE*360 is the angular separation between the edges of the cut after the cone is opened onto the plane, as measured across the surface of the flattened cone. If (RLAT,RLON) is a point to be projected, then the formulas

$$R = (\tan(45-S*RLAT/2))^{**\text{CONE}}$$
$$U = R*\sin(\text{CONE}*(RLON-CLON))$$
$$V = -S*R*\cos(\text{CONE}*(RLON-CLON))$$

where CLON is the longitude of the central meridian, give the coordinates of the projected point in the u/v plane.

The whole globe projects onto the entire u/v plane minus a wedge with its apex at the origin. This projection is normally used to depict mid-latitude regions of limited extent, for which it is relatively distortion-free. It has the property of preserving angles. See example 1.
Azimuthal Projections

Azimuthal projections map the surface of the earth (or of a single hemisphere of it) onto a u/v plane whose origin touches the earth at the user-specified point (PLAT,PLON). The image may be rotated by a user-specified angle ROTA. The azimuthal projections are generated as follows:

1. **Step 1:** Imagine that the earth is placed behind the u/v plane so that the point at latitude 0 and longitude 0 just touches the plane at the point (0,0), the North Pole is at the top, and the South Pole is at the bottom.

2. **Step 2:** Rotate the earth about its polar axis until the v axis is tangent to the meridian identified by PLON (and that meridian is therefore closest to you).

3. **Step 3:** Rotate the earth, tilting one of the poles directly toward you and the other pole directly away from you, until the point (PLAT,PLON) is at the origin of the u/v plane.

4. **Step 4:** Rotate the earth clockwise through the angle ROTA about a line perpendicular to the u/v plane passing through the point (0,0).

5. **Step 5:** Using lines of projection emanating from a central point within or behind the earth (depending on the projection type), project geographical outlines, parallels, and meridians from the earth's surface onto the u/v plane.

6. **Step 6:** Set up linear scales along the u and v axes. The ranges of u and v depend on the projection type.

7. **Step 7:** Draw a rectangular or elliptical portion of the resulting map.

If A is the angular separation, in degrees, of a point P, to be projected, from the point (PLAT,PLON), SINA is the sine of A, COSA is the cosine of A, and R is the linear distance of the projected point P' from the u/v origin, the various azimuthal projections may be described in terms of the relationship between A and R, as follows:

- **Stereographic:** \( R = \tan\left(\frac{A}{2}\right) = \frac{1 - \cos A}{\sin A} \). As A approaches 180 degrees, R approaches infinity. The entire globe is projected to the entire u/v plane. In practice, distortion becomes great beyond R=2, when A is approximately 127 degrees. The center of the projection is the point on the earth's surface opposite the point of tangency with the projection plane. See examples 2, 4, 5, and 7.

- **Orthographic:** \( R = \sin A \). Points for which A is greater than 90 degrees are treated as invisible. Thus, a hemisphere is projected inside a circle of radius 1. The center of the projection is at infinity; all projection lines are parallel to each other and perpendicular to the u/v plane. See example 5.

- **Lambert equal-area:** \( R = 2\sin A / \sqrt{2(1 + \cos A)} \). As A approaches 180 degrees, R approaches 2. The entire globe is projected into a circle of radius 2. See example 5.

- **Gnomonic:** \( R = \tan A = \sin A / \cos A \). Points for which A is greater than 90 degrees are invisible. Thus, a hemisphere is projected to the entire u/v plane. In practice, distortion becomes great beyond R=2,
when A is approximately 65 degrees. The center of this projection is the center of the earth. See example 5.

- **Azimuthal Equidistant:** \( R = A \cdot \pi / 180 \). As A approaches 180 degrees, R approaches \( \pi \) (3.1415926...). Thus, the entire globe is projected within a circle of radius \( \pi \). See example 5.

- **Basic satellite-view:** \( R = \sqrt{S^2 \cdot \sin A} / (S \cdot \cos A) \), where 'SA' is the distance from the center of the earth to a satellite above the point (PLAT,PLON), in multiples of the earth's radius. Points for which \( \cos A \) is less than 1/SA are invisible. The portion of the earth's surface which would be visible from the satellite is projected inside a circle of radius 1. The center of the projection is at the satellite's position. As the satellite moves further and further out, the basic satellite-view projection approaches the orthographic projection. See example 5 and the final example.

The basic satellite-view projection gives a view of the earth as seen from a satellite looking straight down toward the center of the earth. Two other user-settable parameters, called 'S1' and 'S2', are available; they affect the satellite-view projection as follows: S1 measures the angle between the line to the center of the earth and the line of sight of the satellite (the line to which the projection plane is perpendicular). The default value of S1 is zero, which gives the basic satellite view. If S1 is non-zero, S2 specifies the angle from the positive u axis of the basic satellite view counter-clockwise to the line OP, where O is the origin of the basic view and P is the projection (a single point), on the basic view, of the desired line of sight from the satellite. When S1 and S2 are used, the part of the earth projected is the same as for the basic satellite-view projection, but the part of the u/v plane covered by the projection is an ellipse with its major axis at an angle S2 to the u axis. See example 6.

### Cylindrical Projections
Cylindrical projections map the surface of the earth onto a cylinder which is tangent to the earth along a great circle passing through the user-specified point (PLAT,PLON) and tilted at a user-specified angle ROTA. The cylinder is cut along a line parallel to its axis and unrolled onto the plane. The cylindrical projections are generated as follows:

- **Step 1:** Imagine that the earth is placed behind the u/v plane so that the point at latitude 0 and longitude 0 just touches the plane at the point (0,0), the North Pole is at the top, and the South Pole is at the bottom.

- **Step 2:** Rotate the earth about its polar axis until the v axis is tangent to the meridian identified by PLON (and that meridian is therefore closest to you).

- **Step 3:** Rotate the earth, tilting one of the poles directly toward you and the other pole directly away from you, until the point (PLAT,PLON) is at the origin of the u/v plane.

- **Step 4:** Rotate the earth clockwise through the angle ROTA about a line perpendicular to the u/v plane passing through the point (0,0).
- Step 5: Wrap the u/v plane around the globe to form a cylinder, with the u axis touching the earth along a great circle.

- Step 6: Using a technique dependent on the projection type, project geographical outlines, parallels, and meridians outward from the earth's surface onto the cylinder. See below.

- Step 7: Cut the cylinder along a line parallel to its axis and opposite the point (0,0).

- Step 8: Unwrap the cylinder again.

- Step 9: Set up linear scales along the u and v axes. The ranges of u and v depend on the projection type.

- Step 10: Draw a rectangular or elliptical portion of the resulting map.

What happens in step 6 above will be described for each of the three types of cylindrical projections provided by EZMAP in the simple case where PLAT, PLON, and ROTA are all zero. Let RLAT and RLON be the latitude and longitude, in degrees, of a point to be projected; RLON must lie between -180 and +180. (If PLAT, PLON, and/or ROTA are non-zero, one must substitute for RLAT and RLON a pseudo-latitude and a pseudo-longitude computed from the real latitude and longitude; this is left as an exercise for the devotee of spherical trigonometry.) The cylindrical projections may then be described as follows:

- Cylindrical Equidistant: U and V are computed using the equations

  \[
  U = \text{RLON} \\
  V = \text{RLAT}
  \]

  The entire globe is projected into a rectangle in the u/v plane. U ranges from -180 to +180, V from -90 to +90. See example 5.

- Mercator: U and V are computed using the equations

  \[
  U = \text{RLON} \times \pi / 180 \quad \text{(where} \pi = 3.14159 \ldots) \\
  V = \text{ALOG(COT(45 - RLAT/2))}
  \]

  The entire globe is projected into an infinite rectangle in the u/v plane. U ranges from -\pi to +\pi, V from -\infty to +\infty. In practice, distortion becomes unacceptable for latitudes within 5 degrees of the North or South Pole. See examples 3, 4, 5, 8, and 9.

- Mollweide-type: The projection is not a true Mollweide. U and V are computed using the equations

  \[
  U = \text{RLON} / 90 \\
  V = \text{COS}(90 - \text{RLAT})
  \]

  The entire surface of the globe is projected into an ellipse. U ranges from -2 to +2, V from -1 to +1. See example 5.
When an error occurs during a call to an EZMAP routine, the routine SETER is called; by default, it prints an informative error message and STOPs. To recover from errors, insert

```
CALL ENTSR (IDUM,1)
```

at the beginning of your program. This suppresses the STOP in SETER, so that your program gets control back. Then, after each call to an EZMAP routine, insert code like the following:

```
IF (NERRO(IERR).NE.O) THEN
  CALL EPRIN
  CALL ERROF
END IF
```

The value of the function NERRO (and of its argument IERR) is non-zero if and only if SETER has been called. EPRIN prints the error message (which has not yet been done). ERROF turns off an error flag in SETER, telling it that the error has been processed and that execution is to continue. EZMAP's own error flag ("ER") is not cleared; it remains set until the next successful call to MAPINT, to prevent problems in other EZMAP routines.

SETER, ENTSR, NERRO, EPRIN, and ERROF are FORTRAN 77 versions of the PORT routines SETERR, ENTSRC, NERROR, EPRINT, and ERROFF, which are described in detail in the PORT manual.

Possible error-flag values are as follows:

1. MAPGTC  UNKNOWN PARAMETER NAME xx
2. MAPGTI  UNKNOWN PARAMETER NAME xx
3. MAPGTL  UNKNOWN PARAMETER NAME xx
4. MAPGTR  UNKNOWN PARAMETER NAME xx
5. MAPINT  ATTEMPT TO USE NON-EXISTENT PROJECTION
6. MAPINT  ANGULAR LIMITS TOO GREAT
7. MAPINT  MAP HAS ZERO AREA
8. MAPINT  MAP LIMITS INAPPROPRIATE
9. MAPROJ  UNKNOWN PROJECTION NAME xx
10. MAPSET  UNKNOWN MAP AREA SPECIFIER xx
11. MAPSTC  UNKNOWN OUTLINE NAME xx
12. MAPSTC  UNKNOWN PARAMETER NAME xx
13. MAPSTI  UNKNOWN PARAMETER NAME xx
14. MAPSTL  UNKNOWN PARAMETER NAME xx
15. MAPSTR  UNKNOWN PARAMETER NAME xx
16. MAPTRN  ATTEMPT TO USE NON-EXISTENT PROJECTION
17. MAPIO  OUTLINE DATASET IS UNREADABLE
18. MAPIO  EOF ENCOUNTERED IN OUTLINE DATASET
19. MAPOPS  ARGUMENTS ARE INCORRECT
20. MAPRST  ERROR ON READ
21. MAPRST  EOF ON READ
22. MAPSAV  ERROR ON WRITE
IMPLEMENTATION

The EZMAP code is written in FORTRAN 77; portability has been a prime concern.

The binary outline dataset may be generated by running the program

```
PROGRAM CONVRT
DIMENSION FLIM(4), PNTS(200)
CHARACTER*80 CHAR
REWIND 1
REWIND 2
DO 1 I=1,10
READ (1,4) CHAR
CONTINUE
1 CONTINUE
2 READ (1,5,END=3) NPTS, IGID, IDLS, IDRS, (FLIM(I), I=1,4)
   IF (NPTS.GT.1) READ (1,6,END=3) (PNTS(I), I=1,NPTS)
   WRITE (2) NPTS, IGID, IDLS, IDRS, (FLIM(I), I=1,4), (PNTS(I), I=1,NPTS)
GO TO 2
3 STOP
4 FORMAT (A80)
5 FORMAT (A14, 4F8.3)
6 FORMAT (10F8.3)
END
```

with the PORTLIB file EZMAPCDT assigned to unit 1. The output file, on unit 2, contains the binary outline data to be used by EZMAP. The EZMAP routine MAPIO (which see) must then be modified, in a manner dependent on the local system, to access and read this file.

The default values of EZMAP parameters, set by the block data routine MAPBD, should be reviewed. The package has been tuned for NCAR's DICOMED D48 film recorders.

HISTORY

About 1963, R. L. Parker of UCSD wrote the original code, called SUPERMAP, using outline data generated by Hershey. This was adapted for use at NCAR by Lee, in 1968. Revisions occurred in January, 1969, and in May, 1971. The code was put in standard NSSL form in October, 1973. Further revisions occurred in July, 1974, in August, 1976, and in July, 1978. In late 1984 and early 1985, the code was heavily revised to achieve compatibility with FORTRAN 77 and GKS, to remove errors, to augment the outline datasets, and to add enough controls to make user modification of the code unnecessary. In 1987, certain changes were made as part of preparing to introduce a solid-color routine called MAPLAM. The routine MAPEOS was removed and a routine MAPEOD implemented in its place. Intensity parameters 'I1', 'I2', etc., were removed and the color-index parameters 'C1', 'C2', etc. implemented. Cicely Ridley, Jay Chalmers, and Dave Kennison (the current curator) have all had a hand in the creation of this package.

I/O

Graphical output is generated by calls to the routines FRSTD, VECTD, PWRT, and POINTS.

Outline data is read from a FORTRAN logical unit. On the CRAY computers, this unit is automatically assigned and connected to the proper dataset. It is up to the implementor of EZMAP to make similar arrangements on other systems.
ACCURACY The definition of a map drawn by EZMAP is limited by two factors: the resolution of the outline data and the resolution of the graphics device.

Data points in the continental outlines are about one degree apart and the coordinates are accurate to .01 degree. Data points in U.S. state outlines are about .05 degrees apart and the coordinates are accurate to .001 degree. Both the spacing and the accuracy of the international boundaries falls somewhere between these two extremes.

NCAR's DICOMED D48, for which the package has been tuned, has 15-bit coordinate registers, but an effective resolution of at most 1 in 4096 in both x and y.

TIMING The time required to draw a single map is highly variable and depends on how certain parameters are set.

The March 1985, update made EZMAP run significantly slower. This was principally because the default resolution was increased from a value suitable for the dd80 to a value suitable for the DICOMED. Users who are concerned about this may increase the values of the parameters 'MV' and/or 'DD' (see the description of MAPSTx) to improve the speed (at the expense of plot quality, of course).


Parker, R.L., "2UCSD SUPERMAP: World Plotting Package."


ROUTINES This section describes, in alphabetical order, all of the user-callable routines in the EZMAP package.

MAPDRW

Purpose Draws the complete map described by the current values of the EZMAP parameters.

Usage Just call it. MAPDRW calls MAPINT (if required), MAPGRD, MAPLBL, and MAPLOT, in that order. The user may wish to call these routines directly.

Arguments None.
MAPEOD (NOUT,NSEG,IDLS,IDRS,NPTS,PNTS)

Purpose
To examine each outline-dataset segment.

Usage
CALL MAPEOD (NOUT,NSEG,IDLS,IDRS,NPTS,PNTS)

MAPEOD is called by EZMAP to examine each segment in an outline dataset just before it is plotted. The default version does nothing. A user-supplied version may cause selected segments to be deleted (to reduce the clutter in northern Canada, for example). See examples 3, 5, and 9.

Arguments
INTEGER NOUT,NSEG,IDLS,IDRS,NPTS; REAL PNTS(NPTS).

NOUT is the number of the outline dataset from which the segment comes, as follows:

NOUT Dataset to which segment belongs
1 'CO' - Continental outlines only.
2 'US' - U.S. state outlines only.
3 'PS' - Continental, U.S. state, and international outlines.
4 'PO' - Continental and international outlines.

NSEG is the number of the segment within the outline dataset. The maps in example 9 show segment numbers for the outline dataset 'CO'; the program may be modified to produce maps showing segment numbers for any outline dataset.

IDLS and IDRS are identifiers for the areas to the left and right, respectively, of the segment. (Left and right are defined from the standpoint of a viewer standing at point 1 of the segment and looking toward point 2.) For a complete list of area identifiers, see the section entitled "AREA IDENTIFIERS".

NPTS, on entry, is the number of points defining the outline segment. NPTS may be zeroed by MAPEOD to suppress plotting of the segment on the map.

PNTS is an array of coordinates. PNTS(1) and PNTS(2) are the latitude and longitude of the first point, PNTS(3) and PNTS(4) the latitude and longitude of the second point, ..., PNTS(NPTS-1) and PNTS(NPTS) the latitude and longitude of the last point. All values are in degrees. Longitudes are all between -180 and +180; no segment crosses the meridian at -180 (+180) degrees.

MAPFST (RLAT,RLON)

Purpose
Draws lines on the map produced by a call to MAPDRW - used in conjunction with MAPVEC.
**Usage**

The statement

```
CALL MAPFST (RLAT,RLON)
```

is equivalent to the statement

```
CALL MAPIT (RLAT,RLON,0)
```

See the description of MAPIT.

**Arguments**

REAL RLAT,RLON.

RLAT and RLON are defined as for MAPIT (see MAPIT).

---

**MAPGRD**

**Purpose**

Draws a grid.

**Usage**

The statement

```
CALL MAPGRD
```

draws a grid consisting of lines of latitude (parallels) and lines of longitude (meridians). If EZMAP needs initialization or if the error flag 'ER' is non-zero or if the parameter 'GR' is less than or equal to zero, MAPGRD does nothing.

**Arguments**

None.

---

**MAPGTx (WHCH,xVAL)**

**Purpose**

Gets the current value of a specified EZMAP parameter.

**Usage**

Use one of the four statements

```
CALL MAPGTC (WHCH,CVAL)
CALL MAPGTI (WHCH,IVAL)
CALL MAPGTL (WHCH,LVAL)
CALL MAPGTR (WHCH,RVAL)
```

depending on whether the value to be retrieved is of type character, integer, logical, or real. The values of some parameters may be retrieved in more than one way. For example, the value of the initialization flag may be retrieved as a logical quantity (.TRUE. or .FALSE.) or as an integer (non-zero or zero).
Arguments

CHARACTER*2 WHCH, CVAL; INTEGER IVAL; LOGICAL LVAL; REAL RVAL.

WHCH is a character string specifying which parameter to get. Only the first two characters of the string are examined. Possible values are listed in the table below and in the table which is part of the description of MAPSTx.

xVAL (CVAL, IVAL, LVAL, or RVAL) receives the value of the parameter specified by WHCH - of type character, integer, logical, or real, respectively.

All of the parameters listed in the discussion of MAPSTx may be retrieved. The following may also be retrieved:

<table>
<thead>
<tr>
<th>WHCH</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARea</td>
<td>C</td>
<td>The value of the map limits specifier JTLS from the last call to MAPSET. The default value is 'MA'.</td>
</tr>
<tr>
<td>EError</td>
<td>I</td>
<td>The current value of the error flag. The default value is zero (no error).</td>
</tr>
<tr>
<td>INitialize</td>
<td>I,L</td>
<td>Initialization flag. If true (non-zero), it says that EZMAP is in need of initialization (by a CALL MAPINT). The default value is true (non-zero).</td>
</tr>
<tr>
<td>PProjection</td>
<td>C</td>
<td>The value of the projection specifier JPRJ from the last call to MAPROJ. The default value is 'CE'.</td>
</tr>
<tr>
<td>PN</td>
<td>I,R</td>
<td>The value of PLON from the last call to MAPROJ. The default value is zero.</td>
</tr>
<tr>
<td>PT</td>
<td>I,R</td>
<td>The value of PLAT from the last call to MAPROJ. The default value is zero.</td>
</tr>
<tr>
<td>Pn</td>
<td>I,R</td>
<td>&quot;n&quot; is an integer from 1 to 8, inclusive. Retrieves values from the call to MAPSET. P1 through P4 specify PLM1(1), PLM2(1), PLM3(1), and PLM4(1), while P5 through P8 specify PLM1(2), PLM2(2), PLM3(2), and PLM4(2). Default values are all zero.</td>
</tr>
<tr>
<td>RObtation</td>
<td>I,R</td>
<td>The value of ROTA from the last call to MAPROJ. The default value is zero.</td>
</tr>
<tr>
<td>XLeft</td>
<td>R</td>
<td>The parameters XLOW, XROW, YBOW, and YTOW from the last call to MAPPOS. Defaults are .05, .95, .05, and .95, respectively.</td>
</tr>
<tr>
<td>XRight</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>YBottom</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>YTop</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>
MAPINT

Purpose Initialization.
Usage The statement

    CALL MAPINT

initializes the EZMAP package. This is required initially and again after a call to any of the routines MAPPOS, MAPROJ, or MAPSET. The flag 'IN', which may be retrieved by a call to MAPGTI or MAPGTL, indicates whether or not initialization is required at a given time. As of now (April, 1987), no parameter change by means of a call to MAPSTx forces re-initialization; only calls to MAPPOS, MAPROJ, and MAPSET do.

Arguments None.

MAPIQ

Purpose Terminate a string of calls to MAPIT.
Usage The statement

    CALL MAPIQ

flushes MAPIT's buffers. It is particularly important that this be done before a STOP or a CALL FRAME and before changing the color index, dash pattern, etc.

Arguments None.

MAPIT (RLAT, RLON, IFST)

Purpose Draws lines on a map.
Usage

    CALL MAPIT (RLAT,RLON,IFST)

MAPIT is used to draw lines on the map, both by EZMAP itself and, if desired, by the user. MAPIT attempts to omit non-visible portions of lines and to handle "cross-over" - a jump from one end of the map to the other caused by the projection's having slit the globe along some half of a great circle and laid it open with the two sides of the slit at opposite ends of the map. Cross-over can occur on cylindrical and conical projections; MAPIT handles it gracefully on the former and not so well on the latter.
The EZMAP parameter 'DL' determines whether MAPIT draws solid lines or dotted lines. Dotted lines are drawn using calls to POINTS. Solid lines are drawn using calls to DASHD, FRSTD, and VECTD.

The parameters 'DD' and 'MV' also affect MAPIT's behavior. See the description of these parameters in the description of the routine MAPSTx.

A sequence of calls to MAPIT should be followed by a call to MAPIQ (see MAPIQ above) to flush its buffers before a STOP, a CALL FRAME, or a call to change the color index.

Points in two contiguous pen-down calls to MAPIT should not be far apart on the globe.

**Arguments**  
REAL RLAT,RLON.

RLAT and RLON specify the latitude and longitude of a point to which the "pen" is to be moved. Both are given in degrees. RLAT must be between -90. and +90., inclusive; RLON must be between -540. and +540., inclusive.

IFST is 0 to do a "pen-up" move, 1 to do a "pen-down" move only if the distance from the last point to the new point is greater than 'MV' plotter units, and 2 or greater to do a "pen-down" move regardless of the distance from the last point to the new one.

**MAPLBL**

**Purpose**  
Labels the map.

**Usage**  
The statement

```
CALL MAPLBL
```

if the parameter 'LA' is set appropriately, labels the international date line (ID), the equator (EQ), the Greenwich Meridian (GM), and the poles (NP and SP), and, if the parameter 'PE' is set appropriately, draws the perimeter of the map. If EZMAP needs initialization or if the error flag 'ER' is set, MAPLBL does nothing.

**Arguments**  
None.

**MAPLOT**

**Purpose**  
Draws geographical outlines.

**Usage**  
The statement

```
CALL MAPLOT
```
draws the continental and/or international and/or U.S. state outlines selected by the EZMAP parameter 'OU'; the parameter 'DO' determines whether solid lines or dotted lines are used. If EZMAP currently needs initialization or if the error flag 'ER' is set, MAPLOT does nothing.

**Arguments** None.

**MAPPOS (XLOW,XROW,YBOW,YTOW)**

**Purpose** Positions the map on the plotter frame.

**Usage** The statement

```
CALL MAPPOS (XLOW,XROW,YBOW,YTOW)
```

sets four EZMAP parameters specifying the position of a window in the plotter frame within which maps are to be drawn. Each of the arguments is between 0. and 1., inclusive, and specifies the position of one edge of the window, as a fraction of the distance from left to right, or from bottom to top, across the window. The map is centered within the window and made as large as possible, but maintains its intrinsic shape (aspect ratio).

**Arguments** REAL XLOW,XROW,YBOW,YTOW.

- **XLOW** specifies the position of the left edge of the window. The default is .05.
- **XROW** specifies the position of the right edge of the window. The default is .95.
- **YBOW** specifies the position of the bottom edge of the window. The default is .05.
- **YTOW** specifies the position of the top edge of the window. The default is .95.

**MAPROJ (JPRJ,PLAT,PLON,ROTA)**

**Purpose** Set the projection to be used.

**Usage** The statement

```
CALL MAPROJ (JPRJ,PLAT,PLON,ROTA)
```

sets EZMAP parameters specifying the projection to be used for subsequent maps.
**Arguments**

CHARACTER*2 JPRJ; REAL PLAT, PLON, ROTA.

**JPRJ** is a character variable defining the desired projection type, as follows:

The conic projection:

'LC' - Lambert conformal conic with two standard parallels.

The azimuthal projections:

'ST' - Stereographic.

'OR' - Orthographic. The EZMAP parameter 'SA' will be zeroed. See the note below.

'LE' - Lambert equal area.

'GN' - Gnomonic.

'AE' - Azimuthal equidistant.

'SV' - Satellite-view. If the EZMAP parameter 'SA' is less than or equal to 1., it will be reset to 6.631 (the value for a satellite in a geosynchronous orbit). See the note below.

The cylindrical projections:

'CE' - Cylindrical equidistant.

'ME' - Mercator.

'MO' - Mollweide-type.

Note: The orthographic and satellite-view projections have the same internal identifier. The EZMAP parameter 'SA' determines which will be used. If a call to MAPROJ selecting one or the other is followed by a call to MAPSTR resetting 'SA', it may have the effect of causing the other to be used. See the description of 'SA', below, in the description of the routine MAPSTx.

**PLAT**, **PLON**, and **ROTA** are angular quantities, in degrees. How they are used depends on the value of JPRJ, as follows:

- If JPRJ is not equal to 'LC': PLAT and PLON define the latitude and longitude of the pole of the projection - the point on the globe which is to project to the origin of the u/v plane. PLAT must be between -90. and +90., inclusive, positive in the northern hemisphere, negative in the southern. PLON must be between -180. and +180., inclusive, positive to the east, and negative to the west, of Greenwich. ROTA is the angle between the v axis and north at the origin. It is taken to be positive if the angular movement from north to the v axis is counterclockwise, negative otherwise. If the origin is at the north pole, "north" is considered to be in the direction of PLON+180. If the origin is at the south pole, "north" is considered to be in the direction of...
PLON. For the cylindrical projections, the axis of the projection is parallel to the $v$ axis.

- If JPRJ is equal to 'LC': PLON defines the central meridian of the projection, and PLAT and ROTA define the two standard parallels. If PLAT and ROTA are equal, a simpler conic projection, with one standard parallel, is used.

For more detailed descriptions of the projections, see the section called "PROJECTIONS".

**MAPRS**

**Purpose**
Recalls SET.

**Usage**
The statement

```
CALL MAPRS
```

repeats the SET call last done by the routine MAPINT. This might be used when user lines are to be plotted over a map generated in a different overlay (e.g., using a flash buffer), and when the system plot package does not reside in an outer overlay.

**Arguments**
None.

**MAPRST (IFNO)**

**Purpose**
Restores the state of EZMAP saved by an earlier call to MAPSAV.

**Usage**
The statement

```
CALL MAPRST (IFNO)
```

restores EZMAP to a previously saved state (frequently the default state) by reading, from the unit specified by IFNO, values of all user-settable parameters, and then executing MAPINT.

**Arguments**
INTEGER IFNO.

IFNO is the number of a unit from which a single unformatted record is to be read. It is the user's responsibility to position this unit. MAPRST does not rewind it, either before or after reading the record.
MAPSAV (IFNO)

**Purpose**  
Saves the current state of EZMAP for later restoration by MAPRST.

**Usage**  
The statement

```
CALL MAPSAV (IFNO)
```

saves the current state of EZMAP (frequently the default state) by writing, on the unit specified by IFNO, the current values of all the user-settable parameters.

**Arguments**  
INTEGER IFNO.

IFNO is the number of a unit to which a single unformatted record is to be written. It is the user's responsibility to position this unit. MAPSAV does not rewind it, either before or after writing the record.

MAPSET (JLTS,PLM1,PLM2,PLM3,PLM4)

**Purpose**  
To specify the rectangular portion of the u/v plane to be drawn.

**Usage**  
The statement

```
CALL MAPSET (JLTS,PLM1,PLM2,PLM3,PLM4)
```

specifies what portion of the u/v plane is to be plotted.

**Arguments**  
CHARACTER*2 JLTS; REAL PLM1(2),PLM2(2),PLM3(2),PLM4(2).

JLTS is a character string specifying how the limits of the map are to be chosen. There are five possibilities, as follows:

- **JLTS='MA' (MAXIMUM)**. The maximum useful area produced by the projection is plotted. PLM1, PLM2, PLM3, and PLM4 are not used.

- **JLTS='CO' (CORNERS)**. The points (PLM1,PLM2) and (PLM3,PLM4) are to be at opposite corners of the map. PLM1 and PLM3 are latitudes, in degrees. PLM2 and PLM4 are longitudes, in degrees. If a cylindrical projection is being used, the first point should be on the left edge of the map and the second point on the right edge; otherwise, the order makes no difference.

- **JLTS='PO' (POINTS)**. PLM1, PLM2, PLM3, and PLM4 are two-element arrays giving the latitudes and longitudes, in degrees, of four points which are to be on the edges of the rectangular map. If a cylindrical projection is being used, the first point should be on the left edge and the second point on the right edge; otherwise, the order makes no difference.

- **JLTS='AN' (ANGLES)**. PLM1, PLM2, PLM3, and PLM4 are positive angles, in degrees, representing angular distances from a point on the
map to the left, right, bottom, and top edges of the map, respectively. For most projections, these angles are measured with the center of the earth at the vertex and represent angular distances from the point which projects to the origin of the u/v plane; on a satellite-view projection, they are measured with the satellite at the vertex and represent angular deviations from the line of sight. Angular limits are particularly useful for polar projections and for the satellite-view projection; they are not appropriate for the Lambert conformal conic and an error will result if one attempts to use JLTS='AN' with JPRJ='LC'.

- JLTS='LI' (LIMITS). PLM1, PLM2, PLM3, and PLM4 specify the minimum value of u, the maximum value of u, the minimum value of v, and the maximum value of v, respectively. Knowledge of the projection equations is necessary in order to use this option correctly.

MAPSTx (WHCH,xVAL)

Purpose To set the values of certain EZMAP parameters.

Usage Use one of the four statements

CALL MAPSTC (WHCH,CVAL)
CALL MAPSTI (WHCH,IVAL)
CALL MAPSTL (WHCH,LVAL)
CALL MAPSTR (WHCH,RVAL)

depending on whether the value to be set is of type CHARACTER, INTEGER, LOGICAL, or REAL. Some parameters may be set in more than one way. For example, the parameter 'GR', which specifies the grid spacing, may be given the value 10.0 in either of two ways:

CALL MAPSTI ('GR',10)
CALL MAPSTR ('GR',10.)

The flag which controls dotting of outlines may be turned on using either of these calls:

CALL MAPSTI ('DO',1)
CALL MAPSTL ('DO',.TRUE.)

The important point to remember is that the last character of the routine name implies the type of its second argument.

Arguments CHARACTER*2 CVAL; INTEGER IVAL; LOGICAL LVAL; REAL RVAL.

WHCH is a character string specifying the parameter to be set. Only the first two characters of this string are examined. See the table below.
xVAL (CVAL, IVAL, LVAL, or RVAL) contains the value to be given to the parameter specified by WHCH - of type character, integer, logical, or real, respectively.

Following is a list of all the parameters which may be set in this way.

<table>
<thead>
<tr>
<th>WHCH</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cn</td>
<td>I</td>
<td>&quot;n&quot; is an integer between 1 and 7. Each &quot;Cn&quot; specifies the color index of some part of the map. The user must do the calls to GKS routines or to SPPS routines to define the color indices. All default values are zero, indicating no change in color index from one part of the map to another.</td>
</tr>
<tr>
<td>n</td>
<td>Use</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>perimeter</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>grid</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>labels</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>limbs</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>continents</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>U.S. states</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>countries</td>
<td></td>
</tr>
<tr>
<td>DASHpattern</td>
<td>I</td>
<td>Dashed-line pattern for the grids. A 16-bit quantity. The default is 21845 (octal 52525 or binary 0101010101010101).</td>
</tr>
<tr>
<td>DD</td>
<td>I,R</td>
<td>Distance between dots along a dotted line drawn by MAPIT. The default value is 12 (out of 4096. See 'RE', below).</td>
</tr>
<tr>
<td>DL</td>
<td>I,L</td>
<td>If true (non-zero), user calls to MAPIT draw dotted lines. The default is false (zero). Lines drawn by MAPIT are solid or dashed, depending on the current state of the DASHCHAR package. 'DL' may be reset by a user version of MAPUSR to change the way in which the perimeter, the grid, the limb lines, and the outlines are drawn.</td>
</tr>
<tr>
<td>DOT</td>
<td>I,L</td>
<td>If true (non-zero), outlines are dotted. The default is false (zero). Outlines are solid.</td>
</tr>
<tr>
<td>WHCH</td>
<td>Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ELLiptical</td>
<td>I,L</td>
<td>If true (non-zero), only that part of the map which falls inside an ellipse inscribed within the normal rectangular perimeter is drawn. This is particularly appropriate for use with azimuthal projections and angular limits specifying a square, in which case the ellipse becomes a circle, but it will work for any map. The default value is false (zero).</td>
</tr>
<tr>
<td>GD</td>
<td>R</td>
<td>The distance between points used to draw the grid, in degrees. The default value is 1. User values must fall between .001 and 10.</td>
</tr>
<tr>
<td>GRid</td>
<td>I,R</td>
<td>The desired grid spacing. A zero suppresses the grid. The default is 10 degrees.</td>
</tr>
<tr>
<td>LAbel</td>
<td>I,L</td>
<td>If true (non-zero), label the meridians and the poles. The default is true (non-zero).</td>
</tr>
<tr>
<td>LS</td>
<td>I</td>
<td>Controls label size. A character width, to be used in a call to PWRIT. The default value is 1, which gives a character width of 12 plotter units.</td>
</tr>
<tr>
<td>MV</td>
<td>I,R</td>
<td>Minimum vector length for MAPIT. A point closer to the previous point than this is omitted. The default value is 4 (out of 4096. See 'RE', below).</td>
</tr>
<tr>
<td>OUtline</td>
<td>C</td>
<td>Says which set of outline data to use. The possible values are 'NO', for no outlines, 'CO', for continental outlines, 'US', for U.S. state outlines, 'PS', for continental outlines plus international outlines plus U.S. outlines, and 'PO', for continental outlines plus international outlines. Default is 'CO'.</td>
</tr>
<tr>
<td>PErime</td>
<td>I,L</td>
<td>If true (non-zero), draw the perimeter. The default is true (non-zero).</td>
</tr>
<tr>
<td>REsolution</td>
<td>I,R</td>
<td>The width of the target plotter, in plotter units. The default value is 4096.</td>
</tr>
<tr>
<td>SAtellite</td>
<td>I,R</td>
<td>If 'SA' is greater than 1., a satellite-view projection replaces the orthographic. The value is the distance of the satellite from the center of the earth, in multiples of the earth's radius. The default value is zero. See also 'S1' and 'S2', below.</td>
</tr>
</tbody>
</table>
WHCH   Type   Meaning
        
S1 and S2 I,R  Used only when 'SA' is greater than 1. Both are angles, in degrees. ‘S1’ measures the angle between the line to the center of the earth and the line of sight (to which the projection plane is perpendicular). If 'S1' is zero, the projection shows the earth as seen by a satellite looking straight down. Call this the "basic view". If 'S1' is non-zero, 'S2' measures the angle from the positive u axis of the basic view to the line OP, where O is the origin of the basic view and P is the projection of the desired line of sight on the basic view. 'S2' is positive if measured counterclockwise.

SR      R  A search radius, in degrees, used by MAPINT in finding the latitude/longitude range of a map. The default value is 1. User values must lie between .001 and 10. Should not be changed except by advice of a consultant.

MAPTRN (RLAT,RLON,UVAL,VVAL)

Purpose  To project points.

Usage  The statement

    CALL MAPTRN (RLAT,RLON,UVAL,VVAL)

is used to find the projection in the u/v plane of a point whose latitude and longitude are known. MAPTRN may be called at any time after EZMAP has been initialized (by calling MAPINT or otherwise).

Arguments  REAL RLAT,RLON,UVAL,VVAL.

RLAT and RLON are the latitude and longitude, respectively, of a point on the globe. RLAT must be between -90. and +90., inclusive; RLON must be between -540. and +540., inclusive.

(UVAL,VVAL) is the projection in the u/v plane of (RLAT,RLON). The units of UVAL and VVAL depend on the projection.

If the point is not projectable, UVAL is returned equal to 1.E12. Note that, if the point is projectable, but outside the boundary of the map, as defined by the last call to MAPSET, its u and v coordinates are still returned by MAPTRN. The user must do the test required to determine if the point is within limits, if that is necessary.
MAPUSR (IPRT)

Purpose
To change the appearance of the various parts of a map.

Usage
EZMAP executes the statement

CALL MAPUSR (IPRT)

just before and just after each portion of a map is drawn. The default version of MAPUSR does nothing.

A user-supplied version of MAPUSR may set/reset the dotting parameter 'DL', the DASHCHAR dash pattern, the color index, etc., so as to achieve a desired effect.

Arguments
INTEGER IPRT.

IPRT, if positive, says that a particular part of the map is about to be drawn, as follows:

IPRT Part
1 Perimeter.
2 Grid.
3 Labels.
4 Limb lines.
5 Continental outlines.
6 U.S. state outlines.
7 International outlines.

If IPRT is negative, it says that drawing of the last part is complete. The absolute value of IPRT will be one of the above values. Changed quantities should be restored.

MAPVEC (RLAT,RLON)

Purpose
Draws lines on the map produced by a call to MAPDRW - used in conjunction with MAPFST.

Usage
The statement

CALL MAPVEC (RLAT,RLON)

is equivalent to the statement

CALL MAPIT (RLAT,RLON,1)

See the description of MAPIT.
Arguments REAL RLAT,RLON.

RLAT and RLON are defined as for MAPIT (see MAPIT above).

SUPCON (RLAT,RLON,UVAL,VVAL)

Purpose An old equivalent of the new routine MAPTRN. Provided for compatibility with earlier versions of EZMAP/SUPMAP. If efficiency is a consideration, bypass this routine and call MAPTRN directly.

Usage The statement

CALL SUPCON (RLAT,RLON,UVAL,VVAL)

is exactly equivalent to the statement

CALL MAPTRN (RLAT,RLON,UVAL,WVAL)

Arguments REAL RLAT,RLON,UVAL,VVAL.

RLAT, RLON, UVAL, and VVAL are defined as for MAPTRN (see MAPTRN above).

SUPMAP (. . .)

Purpose An implementation of the routine from which EZMAP grew.

Usage The statement

CALL SUPMAP (JPRJ,PLAT,PLON,ROTA,PLM1,PLM2,PLM3,PLM4, + JLTS,JGRD,IOUT,IDOT,IERR)

creates a map of a desired portion of the globe, according to a desired projection, with desired outlines drawn in, and with lines of latitude and longitude at desired intervals. An appropriate call to the routine SET is performed, and the routine SUPCON (see SUPSON above) is initialized so that the user may map points of known latitude and longitude to points in the u/v plane and use the u/v coordinates to draw objects on the map produced by SUPMAP.

Arguments INTEGER JPRJ; REAL PLAT,PLON,ROTA,PLM1(2),PLM2(2),PLM3(2),PLM4(2); INTEGER JLTS,JGRD,IOUT,IDOT,IERR.

IABS(JPRJ) defines the projection type, as follows (values less than 1 or greater than 10 are treated as 1 or 10, respectively):
1 Stereographic.
2 Orthographic.
3 Lambert conformal conic.
4 Lambert equal area.
5 Gnomonic.
6 Azimuthal equidistant.
7 Satellite view.
8 Cylindrical equidistant.
9 Mercator.
10 Mollweide-type.

Using the value 2 causes the EZMAP parameter 'SA' to be zeroed. ('SA', if greater than 1., says that a satellite-view projection, rather than an orthographic projection, is to be used, and specifies the distance of the satellite from the center of the earth, in units of earth radii.) Using the value 7 causes 'SA' to be examined. If it has a non-zero value, the value is left alone. If it has a zero value, its value is reset to 6.631, which is about right for a satellite in a geosynchronous equatorial orbit.

The sign of JPRJ, when IOUT is -1, 0, or +1, indicates whether the continental outlines are to be plotted or not. See IOUT, below.

PLAT, PLON, and ROTA define the origin of the projection and its rotation angle and are used in the same way as they would be in a call to the routine MAPROJ (see MAPROJ above).

JLTS, PLM1, PLM2, PLM3, and PLM4 specify the rectangular limits of the map. These arguments are used in the same way as they would be in a call to MAPSET (see MAPSET above), except that JLTS is an integer instead of a character string. IABS(JLTS) may take on the values 1 through 5, as follows:

1 Like JLTS='MA' in a call to MAPSET.
2 Like JLTS='CO' in a call to MAPSET.
3 Like JLTS='LI' in a call to MAPSET.
4 Like JLTS='AN' in a call to MAPSET.
5 Like JLTS='PO' in a call to MAPSET.

At one time, the sign of JLTS specified whether or not a line of text was to be written at the bottom of the plot produced. This line may no longer be written and the sign of JLTS is therefore ignored.

MOD(IABS(JGRD),1000) is the value, in degrees, of the interval at which lines of latitude and longitude are to be plotted. If the given interval is zero, grid lines and labels are not plotted. If JGRD is less than zero, the perimeter is not plotted. Set JGRD to -1000 to suppress both grid lines and perimeter and to +1000 to suppress the grid lines, but leave the perimeter. The value -0 may have a meaning on ones' complement machines, but should be avoided; use -1000 instead.

IOUT has the value 0 to suppress U.S. state outlines, and the value -1 or +1 to plot U.S. state outlines. In both of these cases, the sign of JPRJ
indicates whether continental outlines are to be plotted (JPRJ positive) or not (JPRJ negative). Originally, SUPMAP recognized only these values of IOUT; now, if IOUT is less than 0 or greater than 1, the sign of JPRJ is ignored, and IOUT selects an outline group, as follows:

-2 or less  NO' (no outlines).
  2       CO' (continental outlines).
  3       US' (U.S. state outlines).
  4       PS' (continental outlines plus international outlines plus U.S. state outlines).
  5 or greater  PO' (continental outlines plus international outlines).

At one time, the sign of IOUT specified whether or not a line of text was to be written on the print output. This may no longer be done.

IDOT is a 0 to get continuous outlines, a 1 to get dotted outlines.

IERR is the only output parameter. A non-zero value indicates that an error has occurred. The section "ERROR CONDITIONS" lists the possible values.
This write-up contains the FORTRAN text and Xerox laser printer output from ten programs which use EZMAP to produce maps. These programs illustrate many of the capabilities of EZMAP.

Example 1 shows the United States on a Lambert conformal conic.

Example 2 shows Africa on a stereographic projection, with an elliptical perimeter.

Example 3 shows simplified continental outlines on a Mercator projection.

Example 4 shows the world in three parts, using stereographic projections for the poles and a Mercator projection for the equatorial belt.

Example 5 shows maximal-area plots for all types of EZMAP projections.

Example 6 shows the effect of the rotation angle ROTA on a satellite-view projection.

Example 7 presents a scheme for labeling the meridians on certain types of maps.

Example 8 demonstrates how the routine MAPUSR is used.

Example 9 shows the numbering of segments in the outline dataset 'CO' and presents a program which may be used to obtain such plots for the other outline datasets. This example is somewhat out of date, since the segment numbers are only of use in a user version of the routine called MAPEOD and it is probably better now to use the left and right area identifiers, rather than the segment numbers, in deciding which outline segments to keep and which to omit.

The final example shows how to draw lines, defined by points for which one has the latitudes and longitudes, on a map produced by EZMAP.

Each of the example programs uses a subroutine called BNDARY to draw a boundary line indicating where the edge of the plotter frame is. This subroutine is not a part of AUTOGRAPH or of the NCAR plot package. It is written as follows:

```fortran
SUBROUTINE BNDARY
C
C Routine to draw the plotter-frame edge.
C
CALL PLOTIT ( 0, 0,0)
CALL PLOTIT (32767, 0,1)
CALL PLOTIT (32767,32767,1)
CALL PLOTIT ( 0,32767,1)
CALL PLOTIT ( 0, 0,1)
RETURN
END
```
Example 1  The United States is mapped using a Lambert conformal conic.
PROGRAM EXMPL1

C The program EXMPL1 produces a map of the U.S., using a Lambert
C conformal conic.
C
C Define the label for the top of the map.
C
CHARACTER*37 PLBL
C
DATA PLBL / 'THE U.S. ON A LAMBERT CONFORMAL CONIC' /
C
Open GKS.
C
CALL OPNGKS
C
Set the outline-dataset parameter.
C
CALL MAPSTC ('OU','US')
C
Set the projection-type parameters.
C
CALL MAPROJ ('LC', 30., -100., 45.)
C
Set the limits parameters.
C
CALL MAPSET ('CO', 22.6, -120., 46.9, -64.2)
C
Draw the map.
C
CALL MAPDRW
C
Put the label at the top of the plot.
C
CALL SET (0., 1., 0., 1., 0., 1., 0., 1., 1.)
CALL PWRIT (.5, .925, PLBL, 37, 2, 0, 0)
C
Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
Advance the frame.
C
CALL FRAME
C
Close GKS.
C
CALL CLSGKS
C
Done.
C
STOP
C
END
**Example 2**  A stereographic view of the international boundaries in Africa, jazzed up by using an elliptical frame.

```
PROGRAM EXMPL2
C
C This program produces a nice view of Africa, with an elliptical perimeter.
C
C Define the label for the top of the map.
C
CHARACTER*26 PLBL
```
C DATA PLBL / 'CAN YOU NAME THE COUNTRIES' /
C
C Open GKS.
C
CALL OPNGKS
C
C Use an elliptical perimeter.
C
CALL MAPSTI ('EL',1)
C
C Dot the outlines, using dots a quarter as far apart as the default.
C
CALL MAPSTI ('DO',1)
CALL MAPSTI ('DD',3)
C
C Show continents and international boundaries.
C
CALL MAPSTC ('OU','PO')
C
C Use a stereographic projection.
C
CALL MAPROJ ('ST',0.,0.,0.)
C
C Specify where two corners of the map are.
C
CALL MAPSET ('CO',-38.,-28.,40.,62.)
C
C Draw the map.
C
CALL MAPDRW
C
C Put the label at the top of the plot.
C
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,0.,1.)
CALL PWRT (.5,.975,PLBL,26,2,0,0)
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Advance the frame.
C
CALL FRAME
C
C Close GKS.
C
CALL CLSGKS
C
C Done.
C
STOP
C
END
Example 3  A Mercator projection of the world, using a version of MAPEOD which
scrubs all but the principal land masses (with a tip of the hat to Cicely
Ridley). The left and right area identifiers are used to determine which
segments to keep and which to scrub. Segment-number information, ob-
tained by running the program shown in example 9, could have been used
instead.
PROGRAM EXMPL3
C
C Produce a Mercator projection of the whole globe, using
C simplified continental outlines. See the routine MAPEOD,
C below.
C
C Define the label for the top of the map.
C
.CHARACTER*46 PLBL
C
DATA PLBL/'SIMPLIFIED CONTINENTS ON A MERCATOR PROJECTION'/
C
Open GKS.
C
CALL OPNGKS
C
Turn off the clipping indicator.
C
CALL GSCLIP (0)
C
Draw the map.
C
CALL SUPMAP (9,0.,0.,0.,0.,0.,1,15,2,0,IERR)
C
Put the label at the top of the plot.
C
CALL SET (0.,1.,0.,1.,0.,0.,1.,1.,1)
CALL PWRIT (.5,.975,PLBL,46,2,0,0)
C
Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
Advance the frame.
C
CALL FRAME
C
Close GKS.
C
CALL CLSGKS
C
Done.
C
STOP
C
END
SUBROUTINE MAPEOD (NOUT,NSEG,IDLS,IDRS,NPTS,PNTS)
C
DIMENSION PNTS(*)
C
This version of MAPEOD uses area identifiers for the outline
C dataset 'CO' to suppress all but the major global land masses.
C In honor of Cicely Ridley, the British Isles are included.
C
C Cull the segment if there's no ocean on either side of it ...
IF (IDLS.NE. 2.AND.IDRS.NE. 2) NPTS=0
C or if it's not an edge of any of the desired areas.
C IF (IDLS.NE. 1.AND.IDRS.NE. 1.AND.
+ IDLS.NE. 3.AND.IDRS.NE. 3.AND.
+ IDLS.NE. 11.AND.IDRS.NE. 11.AND.
+ IDLS.NE. 79.AND.IDRS.NE. 79.AND.
+ IDLS.NE. 99.AND.IDRS.NE. 99.AND.
+ IDLS.NE. 104.AND.IDRS.NE.104.AND.
+ IDLS.NE.107.AND.IDRS.NE.107.AND.
+ IDLS.NE.163.AND.IDRS.NE.163 ) NPTS=0
C Done.
C RETURN
C END
Example 4  The world in three parts. A possibly useful way of depicting the entire
globe, using stereographic projections for the poles and a Mercator projec-
tion for the equatorial belt.

OMNIA TERRA IN PARTES TRES

August 1987 239
PROGRAM EXMPL4

C This program produces a single frame, with polar stereographic views of the poles and a Mercator projection of the rest.
C
C Define the label for the top of the map.
C
CHARACTER*26 PLBL

DATA PLBL / 'OMNIA TERRA IN PARTES TRES' /

C Open GKS.
C
CALL OPNGKS
C
C Turn off the clipping indicator.
C
CALL GSCLIP (0)
C
C Set the outline-dataset parameter.
C
CALL MAPSTC ('OU','PO')
C
C Use dotted outlines and move the dots a little closer together than normal.
C
CALL MAPSTI ('DO',1)
CALL MAPSTI ('DD',3)

C Do the Mercator projection of the equatorial belt first.
C
CALL MAPPOS (.05,.95,.05,.5)
CALL MAPROJ ('ME',0.,0.,0.)
CALL MAPSET ('LI',-3.1416,+3.1416,-1.5708,+1.5708)
CALL MAPDRW

C Switch to an elliptical (in this case, circular) boundary.
C
CALL MAPSTI ('EL',1)
C
C Do a polar stereographic view of the North Pole ...
C
CALL MAPPOS (.07,.48,.52,.93)
CALL MAPROJ ('ST',90.,0.,-90.)
CALL MAPSET ('AN',30.,30.,30.,30.)
CALL MAPDRW

C and then a similar view of the South Pole.
C
CALL MAPPOS (.52,.93,.52,.93)
CALL MAPROJ ('ST','-90.,0.,-90.)
CALL MAPSET ('AN',30.,30.,30.,30.)
CALL MAPDRW

C Put the label at the top of the plot.
C
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRIT (.5,.975,PLBL,26,2,0,0)
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Advance the frame.
C
CALL FRAME
C
C Close GKS.
C
CALL CLSGKS
C
C Done.
C
STOP
C
END
Example 5  Intended as a reference. Maximal-area plots of all the possible EZMAP projections.
PROGRAM EXMPL5
C
C The program EXMPL5 produces a single frame with maximal-area
C views of all the EZMAP projections of the globe.
C
C Define the label for the top of the map.
C
CHARACTER*37 PLBL
C
DATA PLBL / 'MAXIMAL-AREA PROJECTIONS OF ALL TYPES' /
C
C Open GKS.
C
CALL OPNGKS
C
C Turn off the clipping indicator.
C
CALL GSCLIP (0)
C
C Set the outline-dataset parameter.
C
CALL MAPSTC ('OU','CO')
C
C Put meridians and parallels every 15 degrees.
C
CALL MAPSTI ('GR',15)
C
C Reduce the label size.
C
CALL MAPSTI ('LS',0)
C
C Lambert conformal conic.
C
CALL MAPPOS (.025,.24375,.63125,.85)
CALL MAPROJ ('LC',30.,0.,45.)
CALL MAPDRW
C
C Stereographic.
C
CALL MAPPOS (.26875,.4875,.63125,.85)
CALL MAPROJ ('ST',0.,0.,0.)
CALL MAPDRW
C
C Orthographic.
C
CALL MAPPOS (.5125,.73125,.63125,.85)
CALL MAPROJ ('OR',0.,0.,0.)
CALL MAPDRW
C
C Lambert equal-area.
C
CALL MAPPOS (.75625,.975,.63125,.85)
CALL MAPROJ ('LE',0.,0.,0.)
CALL MAPDRW
C
August 1987
243
C Gnomonic.
C
CALL MAPPOS (.025,.24375,.3875,.60625)
CALL MAPROJ ('GN',0.,0.,0.)
CALL MAPDRW
C
C Azimuthal equidistant.
C
CALL MAPPOS (.26875,.4875,.3875,.60625)
CALL MAPROJ ('AE',0.,0.,0.)
CALL MAPDRW
C
C Satellite-view.
C
CALL MAPPOS (.5125,.73125,.3875,.60625)
CALL MAPROJ ('SV',0.,0.,0.)
CALL MAPSTR ('SA',2.)
CALL MAPDRW
C
C Mercator.
C
CALL MAPPOS (.75625,.975,.3875,.60625)
CALL MAPROJ ('ME',0.,0.,0.)
CALL MAPDRW
C
C Cylindrical equidistant.
C
CALL MAPPOS (.025,.4875,.13125,.3625)
CALL MAPROJ ('CE',0.,0.,0.)
CALL MAPDRW
C
C Mollweide type.
C
CALL MAPPOS (.5125,.975,.13125,.3625)
CALL MAPROJ ('MO',0.,0.,0.)
CALL MAPDRW
C
C Put the label at the top of the plot ...
C
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRIT (.5,.925,PLBL,37,2,0,0)
C
C and the labels under each sub-plot.
C
CALL PWRIT (.134375,.61875,'LAMBERT CONFORMAL CONIC', + 23,0,0,0)
CALL PWRIT (.378125,.61875,'STEREOGRAPHIC',13,0,0,0)
CALL PWRIT (.621875,.61875,'ORTHOGRAPHIC',12,0,0,0)
CALL PWRIT (.865625,.61875,'LAMBERT EQUAL-AREA',18,0,0,0)
CALL PWRIT (.134375,.375,'GNOMONIC',8,0,0,0)
CALL PWRIT (.378125,.375,'AZIMUTHAL EQUIDISTANT',21,0,0,0)
CALL PWRIT (.621875,.375,'SATELLITE-VIEW',14,0,0,0)
CALL PWRIT (.865625,.375,'MERCATOR',8,0,0,0)
CALL PWRIT (.25625,.11875,'CYLINDRICAL EQUIDISTANT', + 23,0,0,0)
CALL PWRIT (.74375,.11875, 'MOLLWEIDE TYPE',14,0,0,0)
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Advance the frame.
C
CALL FRAME
C
C Close GKS.
C
CALL CLSGKS
C
C Done.
C
STOP
C
END

SUBROUTINE MAPEOD (NOUT,NSEG, IDLS, IDRS,NPTS,PNTS)
C
DIMENSION PNTS(*)
C
This version of MAPEOD uses area identifiers for the outline
C dataset 'CO' to suppress all but the major global land masses.
C
IF (IDLS.NE. 2.AND.IDRS.NE. 2) NPTS=0
C
IF (IDLS.NE. 1.AND.IDRS.NE. 1.AND.
+ IDLS.NE. 3.AND.IDRS.NE. 3.AND.
+ IDLS.NE. 11.AND.IDRS.NE. 11.AND.
+ IDLS.NE. 79.AND.IDRS.NE. 79.AND.
+ IDLS.NE. 99.AND.IDRS.NE. 99.AND.
+ IDLS.NE.104.AND.IDRS.NE.104.AND.
+ IDLS.NE.107.AND.IDRS.NE.107.AND.
+ IDLS.NE.163.AND.IDRS.NE.163 ) NPTS=0
C
C Done.
C
RETURN
C
END
Example 6  Shows the effect of changing the rotation angle ROTA. Note that the satellite is positioned over Kansas.
PROGRAM EXMPL6

C

This program produces a single frame showing satellite views of the globe, each rotated by a different angle.

C

Define the label for the top of the map.

C

CHARACTER*27 PLBL

C

DATA PLBL / 'THE EARTH IS SPINNING, TOTO'/

C

Open GKS.

C

CALL OPNCKS

C

Set the outline-dataset parameter.

C

CALL MAPSTC ('OU', 'PS')

C

Use a satellite-view projection.

C

CALL MAPSTR ('SA', 1.25)

C

Aim the camera 15 degrees away from straight down.

C

CALL MAPSTI ('S1', 15)

C

Turn off the perimeter and reduce the number of grid lines.

C

CALL MAPSTI ('PE', 0)

CALL MAPSTI ('GR', 15)

C

Center the first map over Kansas. Rotate by 0 degrees and look to the upper left.

C

CALL MAPPOS (.05, .475, .525, .95)

CALL MAPROJ ('SV', 38., -98., 0.)

CALL MAPSTI ('S2', 135)

CALL MAPDRW

C

Repeat, but rotate by 90 degrees and look to the upper right.

C

CALL MAPPOS (.525, .95, .525, .95)

CALL MAPROJ ('SV', 38., -98., 90.)

CALL MAPSTI ('S2', 45)

CALL MAPDRW

C

Repeat, but rotate by 180 degrees and look to the lower left.

C

CALL MAPPOS (.05, .475, .05, .475)

CALL MAPROJ ('SV', 38., -98., 180.)

CALL MAPSTI ('S2', -135)

CALL MAPDRW

C

Repeat, but rotate by 270 degrees and look to the lower right.
CALL MAPPOS (.525, .95, .05, .475)
CALL MAPROJ ('SV', 38., -98., 270.)
CALL MAPSTI ('S2', -45)
CALL MAPDRW

C Put the label at the top of the plot ...
C
CALL SET (0., 1., 0., 1., 0., 1., 0., 1., 1)
CALL PWRT (.5, .975, PLBL, 27, 2, 0, 0)

C and the ones below each sub-plot.
C
CALL PWRT (.2625, .5, 'ROTA = 0', 8, 1, 0, 0)
CALL PWRT (.7375, .5, 'ROTA = 90', 9, 1, 0, 0)
CALL PWRT (.2625, .025, 'ROTA = 180', 10, 1, 0, 0)
CALL PWRT (.7375, .025, 'ROTA = 270', 10, 1, 0, 0)

C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY

C Advance the frame.
C
CALL FRAME

C Close GKS.
C
CALL CLSGKS

C Done.
C
STOP

C
END
Example 7  It is difficult to write a general algorithm for labeling the meridians and parallels on any map. The capability has therefore been omitted from EZMAP proper and is left to the user. This example presents a routine which will label the meridians on certain kinds of maps. Similar techniques enable one to label other types of maps.
PROGRAM EXMPL7

C This program produces a stereographic view of the North Pole, C with labeled meridians.
C
C Define the label for the top of the map.
C
CHARACTER*32 PLBL
C
DATA PLBL / 'MERIDIONAL LABELS ON A POLAR MAP' /
C
C Open GKS.
C
CALL OPNGKS
C
C Turn off the clipping indicator.
C
CALL GSCLIP (0)
C
C Move the map a little to provide more room for labels.
C
CALL MAPPOS (.075,.925,.05,.90)
C
C Use an elliptical (circular, in this case) perimeter.
C
CALL MAPSTI ('EL',1)
C
C Show continents and international boundaries.
C
CALL MAPSTC ('OU','PO')
C
C Use a stereographic projection, centered at the North Pole.
C
CALL MAPROJ ('ST',90.,0.,-100.)
C
C Specify the angular distances to the edges of the map.
C
CALL MAPSET ('AN',80.,80.,80.,80.)
C
C Draw the map.
C
CALL MAPDRW
C
C Call a routine to label the meridians. This routine is not C a part of EZMAP; the code is given below.
C
CALL MAPLEBM
C
C Put the label at the top of the plot.
C
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PWRIT (.5,.975,PLBL,32,2,0,0)
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C  Advance the frame.
C
  CALL FRAME
C
  Close GKS.
C
  CALL CLSGKS
C
  Done.
C
  STOP
C
END
SUBROUTINE MAPLBM
C
This routine labels the meridians if and only if the current
projection is azimuthal and centered at one of the poles, a
circular boundary is being used, and the grid increment is an
integral divisor of 180. The routine was not thought general
enough to include in EZMAP itself, but may nevertheless be of
interest to users.
C
Necessary local declarations.
C
  CHARACTER*2 PROJ
  CHARACTER*3 CHRS
  CHARACTER*4 CHLB
C
See if the conditions required for MAPLBM to work are met.
C
The projection must be azimuthal, ...
C
  CALL MAPGTC ('PR',PROJ)
  IF (PROJ.NE.'ST'.AND.PROJ.NE.'OR'.AND.PROJ.NE.'LE'.AND.
      PROJ.NE.'GN'.AND.PROJ.NE.'AE') RETURN
C
the pole latitude must be +90 degrees or -90 degrees, ...
C
  CALL MAPGTR ('PT',PLAT)
  IF (ABS(PLAT).LT.89.9999) RETURN
C
the perimeter must be elliptical, ...
C
  CALL MAPGTI ('EL',IELP)
  IF (IELP.EQ.0) RETURN
C
the values used in the SET call must define a circle, ...
C
  CALL GETSET (FLEW,FREW,FBEW,FTEW,ULEW,UREW,VBEW,VTEW,LNLG)
  ILEW=KFPX(FLEW)
  IREW=KFPX(FREW)
  IBEW=KFPY(FBEW)
  ITEW=KFPY(FTEW)
IF (ULEW+UREW.GT.0.0001.OR.VBEW+VTEW.GT.0.0001) RETURN
IF (ULEW+VTEW.GT.0.0001.OR.VBEW+UREW.GT.0.0001) RETURN

C
C and the grid spacing must be an integral divisor of 180.
C
CALL MAPGTR ('GR',GRID)
IF (AMOD(GRID,1.).NE.0.OR.
+ MOD(180,IFIX(GRID)).NE.0) RETURN

C
C All conditions are satisfied. Label the meridians.
C
C Collect the necessary information.
C
IGRD=GRID
CALL MAPGTR ('PN',PLON)
CALL MAPGTR ('RO',ROTA)
CALL MAPGTI ('LS',ICSZ)
IF (ICSZ.EQ.0) THEN
  ICSZ=8
ELSE IF (ICSZ.EQ.1) THEN
  ICSZ=12
ELSE IF (ICSZ.EQ.2) THEN
  ICSZ=16
ELSE IF (ICSZ.EQ.3) THEN
  ICSZ=24
END IF
WCH=(FLOAT (ICSZ)/FLOAT (IREW-ILEW)) * (UREW-ULEW)
HOCH=(FLOAT (2*ICSZ)/FLOAT(ITEW-IBEW)) * (VTEW-VBEW)
HOLB=HOCH/1.5

C
C Loop on the label values.
C
DO 101 I=-180,179,IGRD

C
C Express the value of the longitude in a nice form.
C
WRITE (CHRS,1001) IABS(I)
NCHS=0
IF (IABS(I).GE.100) THEN
  NCHS=NCHS+1
  CHLB(NCHS:NCHS)=CHRS(1:1)
END IF
IF (IABS(I).GE.10) THEN
  NCHS=NCHS+1
  CHLB(NCHS:NCHS)=CHRS(2:2)
END IF
NCHS=NCHS+1
CHLB(NCHS:NCHS)=CHRS(3:3)
IF (I.GT.-180.AND.I.LT.0) THEN
  NCHS=NCHS+1
  CHLB(NCHS:NCHS)='W'
ELSE IF (I.GT.0.AND.I.LT.180) THEN
  NCHS=NCHS+1
  CHLB(NCHS:NCHS)='E'
END IF
EZMAP

C Compute the width of the label.
C
C
C
C
C
C
C
C
WOLB=FLOAT(NCHS)*WOCH

C Find the angle at which the labeled meridian lies on the plot.
C
IF (PLAT.GT.0.) THEN
   ANGD=FLOAT(I-90)-PLON-ROTA
ELSE
   ANGD=FLOAT(90-I)+PLON-ROTA
END IF

C Reduce the angle to the range from -180 to +180 and find its equivalent in radians.
C
ANGD=ANGD-SIGN(180.,ANGD+180.)
   +SIGN(180.,180.-ANGD)
ANGR=.017453292519943*ANGD

C Figure out where the end of the meridian is.
C
XEND=UREW*COS(ANGR)
YEND=VTEW*SIN(ANGR)

C Extend the meridian a little to make a tick mark.
C
CALL LINE (XEND,YEND,1.015*XEND,1.015*YEND)

C Compute a center position for the label which puts its nearest edge at a fixed distance from the perimeter. First, compute the components (DELX,DELY) of the vector from the center of the label box to the edge nearest the perimeter.
C
IF (ANGD.LT.-179.9999) THEN
   DELX=+0.5*WOLB
   DELY= 0.
ELSE IF (ANGD.LT.-90.0001) THEN
   DELX=+0.5*WOLB
   DELY=+0.5*HOLB
ELSE IF (ANGD.LT.-89.9999) THEN
   DELX= 0.0
   DELY=+0.5*HOLB
ELSE IF (ANGD.LT.-0.0001) THEN
   DELX=-0.5*WOLB
   DELY=+0.5*HOLB
ELSE IF (ANGD.LT.+0.0001) THEN
   DELX=-0.5*WOLB
   DELY= 0.0
ELSE IF (ANGD.LT.+89.9999) THEN
   DELX=-0.5*WOLB
   DELY=-0.5*HOLB
ELSE IF (ANGD.LT.+90.0001) THEN
   DELX= 0.0
   DELY=-0.5*HOLB

August 1987
ELSE IF (ANGD.LT.+179.9999) THEN
  DELX=+0.5*WOLB
  DELY=-0.5*HOLB
ELSE
  DELX=+0.5*WOLB
  DELY= 0.0
END IF

C Then, solve (for FMUL) the following equation:
C
C  SQRT((FMUL*XENI+DELX)**2+(FMUL*YEND+DELY)**2))=1.02*UREW
C
C which expresses the condition that the corner of the box
C nearest the circular perimeter should be at a distance of
C 1.02*(the radius of the perimeter) away from the center of
C the plot.
C
A=XEND*XEND+YEND*YEND
B=2.*(XEND*DELX+YEND*DELY)
C=DELX*DELX+DELY*DELY-1.0404*UREW*UREW

FMUL=(-B+SQRT(B*B-4.*A*C))/(2.*A)

C Draw the label.
C
CALL PWRIT (FMUL*XEND,FMUL*YEND,CHLB,NCHS,ICSZ,0,0)

C End of loop.
C
101 CONTINUE
C
C Done.
C
RETURN
C
C Format
C
1001 FORMAT (1I3)
C
END
Example 8  This example demonstrates how a user-provided routine MAPUSR may be employed to change the way in which portions of a map are drawn.
PROGRAM EXMPL8

C
C Produce a Mercator projection of the whole globe, using a
C version of MAPUSR which dots the grid lines and dashes the
C continental outlines.
C
C Define the label for the top of the map.
C
CHARACTER*30 PLBL

C
DATA PLBL / 'ILLUSTRATING THE USE OF MAPUSR' /

C
Open GKS.
C
CALL OPNGKS
C
C Weird up the projection a little.
C
CALL SUPMAP (9.0.,0.,90.,0.,0.,0.,0.,0.,1,15,2,0,IERR)
C
C Put the label at the top of the plot.
C
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRT (.5,.975,PLBL,30,2,0,0)

C
Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Advance the frame.
C
CALL FRAME
C
C Close GKS.
C
CALL CLSGKS
C
C Done.
C
STOP
C

END

SUBROUTINE MAPUSR (IPRT)

C
This version of MAPUSR forces the grid lines to be dotted and
the outlines to be dashed.
C
C Certain local parameters must be saved from call to call.
C
SAVE INTN,IDTF,IDBD

C
If IPRT is positive, a part is about to be drawn. Save the
C dotted/solid flag and/or the distance between dots and then
C reset them and/or the dash pattern.  

IF (IPRT.GT.0) THEN
  IF (IPRT.EQ.2) THEN
    CALL MAPGTI ('DL', IDTF)
    CALL MAPGTI ('DD', IDBD)
    CALL MAPSTI ('DL', 1)
    CALL MAPSTI ('DD', 3)
  ELSE IF (IPRT.EQ.5) THEN
    CALL MAPGTI ('DL', IDTF)
    CALL MAPSTI ('DL', 0)
    CALL DASHDB (52525B, 0, 0, 0)
  END IF
C
ELSE
  C
  Otherwise, a part has just been drawn. Restore saved settings
  C and/or select a solid dash pattern.
  C
  IF (IPRT.EQ.-2) THEN
    CALL MAPSTI ('DL', IDTF)
    CALL MAPSTI ('DD', IDBD)
  ELSE IF (IPRT.EQ.-5) THEN
    CALL MAPSTI ('DL', IDTF)
    CALL DASHDB (17777B, 0, 0, 0)
  END IF
C
END IF
C
C Done.
C
RETURN
C
END
Example 9  The four plots produced show the numbering of segments in the outline dataset 'CO'. This information may be used to produce simplified continental outlines; see examples 3 and 5 for another way.

The program can produce segment-number plots for the other outline datasets, also.

The algorithm used to prevent numbers from landing on top of one another doesn't prevent them from landing on top of the outlines, which makes some of them hard to read. (The original is considerably better than the reproduction.)

The technique used to break the maps into rectangular pieces and present each piece individually may be useful in its own right.
SEGMENT NUMBERS FOR OUTLINE DATASET CO
PROGRAM EXMPL9
C
C The program EXMPL9 produces a set of plots showing the numbers
C of all the segments in a chosen EZMAP outline dataset. Certain
C variables must be set in data statements at the beginning of
C the program. In each of the seven places marked off by rows of
C dashes, un-comment the first card to do outline dataset 'CO',
C the second to do 'US', the third to do 'PO', and the fourth to
C do 'PS'.
C
C The common block LIMITS communicates values between TESTIT and
C the routines MAPEOD and MOVEIT.
C
COMMON /LIMITS/ ICSZ,ULEW,UREW,VBEW,VTEW,DELU,DELV,NSAV
C
C Select the outline dataset.
C
CHARACTER*2 OUTD
C
C-----------------------
DATA OUTD / 'CO' /
DATA OUTD / 'US' /
DATA OUTD / 'PO' /
DATA OUTD / 'PS' /
C-----------------------
C
C Select the projection type.
C
CHARACTER*2 PROJ
C
C-----------------------
DATA PROJ / 'ME' /
DATA PROJ / 'LC' /
DATA PROJ / 'ME' /
DATA PROJ / 'ME' /
C-----------------------
C
C Select the appropriate values of PLAT, PLON, and ROTA.
C
DATA PLAT,PLON,ROTA / 0., 0., 0. /  
DATA PLAT,PLON,ROTA / 30., -100., 45. / 
DATA PLAT,PLON,ROTA / 0., 0., 0. / 
DATA PLAT,PLON,ROTA / 0., 0., 0. / 
C
C Select the parameter saying how the map limits are chosen.
C
CHARACTER*2 LMTS
C
C-----------------------
DATA LMTS / 'MA' /
DATA LMTS / 'CO' /
DATA LMTS / 'MA' /
DATA LMTS / 'MA' /
C-----------------------
C-----------------------
C
C Select the values to be put in the limits arrays.
C
DIMENSION PL1(2),PL2(2),PL3(2),PL4(2)
C
C---------------------------------------------------------------
DATA PL1(1),PL2(1),PL3(1),PL4(1) / 0., 0., 0., 0. / 
C DATA PL1(1),PL2(1),PL3(1),PL4(1) / 22.6,-120.,46.9,-64.2 / 
C DATA PL1(1),PL2(1),PL3(1),PL4(1) / 0., 0., 0., 0. / 
C DATA PL1(1),PL2(1),PL3(1),PL4(1) / 0., 0., 0., 0./
C---------------------------------------------------------------
C
C---------------------------------------------------------------
DATA PL1(2),PL2(2),PL3(2),PL4(2) / 0., 0 ., 0., 0. / 
C DATA PL1(2),PL2(2),PL3(2),PL4(2) / 0., 0., 0., 0. / 
C DATA PL1(2),PL2(2),PL3(2),PL4(2) / 0., 0., 0., 0. / 
C DATA PL1(2),PL2(2),PL3(2),PL4(2) / 0., 0 ., 0., 0. / 
C---------------------------------------------------------------
C
C Select values determining how the whole map is to be carved
C up into little maps.  ILIM is the number of divisions of the
C horizontal axis, JLIM the number of divisions of the vertical
C axis.  (If all the labels are put on a single map, the result
C is confusing.)
C
C---------------------------------------------------------------
DATA ILIM,JLIM / 2,2 / 
C DATA ILIM,JLIM / 3,2 / 
C DATA ILIM,JLIM / 2,2 / 
C DATA ILIM,JLIM / 6,6 / 
C---------------------------------------------------------------
C
C Define a variable to hold the plot label.
C
CHARACTER*38 PLBL
C
DATA PLBL / 'SEGMENT NUMBERS FOR OUTLINE DATASET XX' / 
C
C Open GKS.
C
CALL OPNGKS
C
C Finish the plot label.
C
PLBL(37:38)=OUTD
C
C Set the character size; 6 is about the smallest usable value.
C
ICSZ=6
C
C Set the outline-dataset parameter.
C
CALL MAPSTC ('OU',OUTD)
C
August 1987 263
C Set the projection-type parameters.
C
CALL MAPROJ (PROJ,PLAT,PLON,ROTA)
C
C Set the limits parameters.
C
CALL MAPSET (LMTS,PL1,PL2,PL3,PL4)
C
C Initialize EZMAP.
C
CALL MAPINT
C
C Retrieve the values with which MAPINT called SET.
C
CALL GETSET (FLEM,FREM,FBEM,FTEM,ULEM,UREM,VBEM,VTEM,LNLG)
ILEM=KFPX (FLEM)
IREM=KFPX (FREM)
IBEM=KFPY (FBEM)
ITEM=KFPY (FTEM)
C
C Now, plot a set of maps which are subsets of the whole map.
C
DO 102 I=1,ILIM
   DO 101 J=1,JLIM
      CALL MAPSET ('LI',
               ULEM+(UREM-ULEM) *FLOAT(I-1)/FLOAT(ILIM),
               UREM+(UREM-UREM)*FLOAT(I  )/FLOAT(ILIM),
               VBEM+(VTEM-VBEM)*FLOAT(J-1)/FLOAT(JLIM),
               VBEM+(VTEM-VBEM)*FLOAT(J  )/FLOAT(JLIM))
C
C Re-initialize EZMAP with the new limits.
C
CALL MAPINT
C
C Retrieve the values with which MAPINT called SET.
C
CALL GETSET (FLEW,FREW,FBEW,FTEW,
               ULEW,UREW,VBEW,VTEW,LNLG)
ILEW=KFPX (FLEW)
IREW=KFPX (FREW)
IBEW=KFPY (FBEW)
ITEM=KFPY (FTEW)
C
C Compute quantities required by MAPEOD and MOVEIT to position
C labels.
C
DELU=3.5* (FLOAT (ICSZ)/FLOAT (IREW-ILEW))* (UREW-ULEW)
DELV=2.0* (FLOAT (ICSZ)/FLOAT (ITEW-IBEW))* (VTEW-VBEW)
NSAV=0
C
C Draw the outlines.
C
CALL MAPLOT
C
C Put a label at the top of the plot.
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRIT (.5,.975,PLBL,38,2,0,0)

C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY

C Advance the frame.
C
CALL FRAME

101 CONTINUE
102 CONTINUE

C Close GKS.
C
CALL CLSGKS

C Done.
C
STOP

END

SUBROUTINE MAPEOD (NOUT,NSEG,IDLS,IDRS,NPTS,PNTS)

DIMENSION PNTS(*)

C This version of MAPEOD marks each segment of the map with its
C segment number. The resulting map may be used to set up other
C versions of MAPEOD which plot selected segments.
C
C The common block LIMITS communicates values between TESTIT and
C the routines MAPEOD and MOVEIT.
C
COMMON /LIMITS/ ICSZ,ULEW,UREW,VBEW,VTEW,DELU,DELV,NSAV

C Define local variables to hold the character-string form of the
C segment number.
C
CHARACTER*4 CSG1,CSG2

C Find out where on the map the center of the segment is.
C
MPTS=NPTS/2+1
CALL MAPTRN (PNTS(2*MPTS-1),PNTS(2*MPTS),UCEN,VCEN)

C If the center is visible, label it.
C
IF (UCEN.GE.ULEW.AND.UCEN.LE.UREW.AND.
+ VCEN.GE.VBEW.AND.VCEN.LE.VTEW) THEN

C Generate a character string representing the value of the
C segment number.
C
August 1987
WRITE (CSG2,1001) NSEG
NFCH=4-IFIX(ALOG10(FLOAT(NSEG)+.5))
NCHS=5-NFCH
CSG1=CSG2(NFCH:4)

C Find out where the two points on either side of the center are.
C
MPTS=MAXO(NPTS/2,1)
CALL MAPTRN (PNTS(2*MPTS-1),PNTS(2*MPTS),UCM1,VCM1)
C
MPTS=MINO(NPTS/2+2,NPTS)
CALL MAPTRN (PNTS(2*MPTS-1),PNTS(2*MPTS),UCP1,VCP1)
C
C Compute the preferred position of the label, with one corner
C of its enclosing box at the center of the segment.
C
ULAB=UCEN-SIGN(.1428*DELU*FLOAT(NCHS), UCM1+UCP1-2*UCEN)
VLAB=VCEN-SIGN(.3333*DELV,VCM1+VCP1-2*VCEN)
C
C Move the label as necessary to avoid its being on top of any
C previous label.
C
CALL MOVEIT (ULAB,VLAB)
C
C Write out the character string and connect it to the segment
C with a straight line.
C
CALL LINE (UCEN,VCEN,
+          ULAB-SIGN(.1428*DELU*FLOAT(NCHS),ULAB-UCEN),
+          VLAB-SIGN(.3333*DELV,VLAB-VCEN))
C
CALL PWRT (ULAB,VLAB,CSG1,NCHS,ICSZ,O,O)
C
END IF
C
C Done.
C
RETURN
C
C Format.
C
1001 FORMAT (I4)
C
END
SUBROUTINE MOVEIT (ULAB,VLAB)
C
C The object of this routine is to avoid putting segment labels
C on top of each other. (ULAB,VLAB), on entry, is the most
C desirable position for a segment label. MOVEIT modifies this
C position as necessary to make sure the label will not be on top
C of any previous label.
C
C The common block LIMITS communicates values between TESTIT and
C the routines MAPEOD and MOVEIT.
C
EZMAP

C ULEW, UREW, VBEW, and VTEW specify the u/v limits of the window in which the map was drawn. DELU and DELV are the minimum allowable distances between segment-label centers in the u and v directions, respectively. NSAV is the number of label positions saved in the arrays USAV and VSAV.

COMMON /LIMITS/ ICSZ, ULEW, UREW, VBEW, VTEW, DELU, DELV, NSAV

C Previous label positions are saved in the arrays USAV and VSAV.

DIMENSION USAV(1000), VSAV(1000)

C USAV and VSAV must maintain their values between calls.

SAVE USAV, VSAV

C Zero the variables which control the generation of a spiral.

IGN1=0
IGN2=2

C Check for overlap at the current position.

101 DO 102 I=1, NSAV
     IF (ABS(ULAB-USAV(I)).LT.DELU.AND.
+     ABS(VLAB-VSAV(I)).LT.DELV) GO TO 103
102 CONTINUE

C No overlap. Save the new label position and return to caller.

NSAV=NSAV+1
IF (NSAV.GT.1000) STOP 1
USAV(NSAV)=ULAB
VSAV(NSAV)=VLAB

RETURN

C Overlap. Try a new point. The points tried form a spiral.

103 IGN1=IGN1+1
     IF (IGN1.LE.IGN2/2) THEN
        ULAB=ULAB+SIGN(DELU,-.5+FLOAT(MOD(IGN2/2,2)))
     ELSE
        VLAB=VLAB+SIGN(DELV,-.5+FLOAT(MOD(IGN2/2,2)))
     END IF
     IF (IGN1.EQ.IGN2) THEN
        IGN1=0
        IGN2=IGN2+2
     END IF
     IF (ULAB.LT.ULEW.OR.ULAB.GT.UREW.OR.
+     VLAB.LT.VBEW.OR.VLAB.GT.VTEW) GO TO 103
     GO TO 101

END
A Final Example On a satellite-view projection of the North Atlantic appears an appropriate message. Note the use of user-drawn dotted lines.
PROGRAM EXAMPLE

C Define a data array.
C
DIMENSION XYCD(224)
C
C Define the centers and the expansion/shrinkage factors for various copies of the curve to be drawn.
C
DIMENSION FLAC(4), FLOC(4), FMUL(4)
C
DATA FLOC(1), FLAC(1), FMUL(1) / -38.1, 32.0, 1.7 /
DATA FLOC(2), FLAC(2), FMUL(2) / -37.9, 32.0, 1.7 /
DATA FLOC(3), FLAC(3), FMUL(3) / -38.0, 31.9, 1.7 /
DATA FLOC(4), FLAC(4), FMUL(4) / -38.0, 32.1, 1.7 /
C
C Open GKS.
C
CALL OPNGKS
C
C Fill the data array.
C
READ 1001, XYCD
C
C Define the altitude of the satellite.
C
CALL MAPSTR ('SA', 2.)
C
C Draw a map of the North Atlantic, as seen by a satellite.
C
CALL SUPMAP (7, 32., -38., 20., 0., 0., 0., 0., 1., -1000, 5.0, IERR)
C
C Force MAPIT to draw dotted lines.
C
CALL MAPSTI ('DL', 1)
CALL MAPSTI ('DD', 3)
C
C Draw some curves.
C
DO 102 I=1, 4
C
IFST=0
C
DO 101 J=1, 112
  IF (XYCD(2*J-1).EQ.0.) THEN
    IFST=0
  ELSE
    FLON=FLOC(I)+FMUL(I)*(XYCD(2*J-1)-15.)
    FLAT=FLAC(I)+FMUL(I)*(XYCD(2*J )-15.)
    CALL MAPIT (FLAT, FLON, IFST)
    IFST=1
  END IF
101 CONTINUE
C
102 CONTINUE
C Dump MAPIT's buffers.
C
   CALL MAPIQ
C
C Draw a boundary around the edge of the plotter frame.
C
   CALL BNDARY
C
C Advance the frame.
C
   CALL FRAME
C
C Close GKS.
C
   CALL CLSGKS
C
C Done.
C
   STOP
C
C Format.
   1001 FORMAT (14E5.0)
C
   END
AREA IDENTIFIERS

There are four EZMAP outline datasets, indexed as follows:

<table>
<thead>
<tr>
<th>Index</th>
<th>Dataset name and contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'CO' - Continental outlines only.</td>
</tr>
<tr>
<td>2</td>
<td>'US' - U.S state outlines only.</td>
</tr>
<tr>
<td>3</td>
<td>'PS' - Continental, U.S. state, and international outlines.</td>
</tr>
<tr>
<td>4</td>
<td>'PO' - Continental and international outlines.</td>
</tr>
</tbody>
</table>

Associated with each outline segment in these datasets are two integers identifying the areas to the left and right of the outline segment. These "area identifiers" may be used, in a user version of the subroutine MAPEOD, to cull unwanted segments from a map. The area identifiers are also used by EZMAPA and AREAS when you are producing a solid-colored map. See the EZMAPA and AREAS documentation for details.

The following table shows the association between area identifiers and the names of the areas they identify. Column 1 contains an outline dataset index, column 2 an area identifier, column 3 a suggested color index for the area (which will ensure that adjacent areas have different colors), and column 4 the name of the area.

<table>
<thead>
<tr>
<th>Index</th>
<th>Dataset name and contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2       Antarctica</td>
</tr>
<tr>
<td>1</td>
<td>1       Oceans</td>
</tr>
<tr>
<td>1</td>
<td>3       Africa/Eurasia</td>
</tr>
<tr>
<td>1</td>
<td>4       USSR - Wrangel Is.</td>
</tr>
<tr>
<td>1</td>
<td>5       Alaska - Atka Is.</td>
</tr>
<tr>
<td>1</td>
<td>6       New Zealand - Chatham Is.</td>
</tr>
<tr>
<td>1</td>
<td>7       Samoa - Savaii</td>
</tr>
<tr>
<td>1</td>
<td>8       Samoa - Upolu</td>
</tr>
<tr>
<td>1</td>
<td>9       Alaska - Saint Lawrence Is.</td>
</tr>
<tr>
<td>1</td>
<td>10      Alaska - Umnak Is.</td>
</tr>
<tr>
<td>1</td>
<td>11      North and South America</td>
</tr>
<tr>
<td>1</td>
<td>12      Alaska - Unalaska Is.</td>
</tr>
<tr>
<td>1</td>
<td>13      Alaska - Nunivak Is.</td>
</tr>
<tr>
<td>1</td>
<td>14      Antarctica - Roosevelt Is.</td>
</tr>
<tr>
<td>1</td>
<td>15      Hawaiian Is. - Niihau</td>
</tr>
<tr>
<td>1</td>
<td>16      Hawaiian Is. - Kauai</td>
</tr>
<tr>
<td>1</td>
<td>17      Hawaiian Is. - Oahu</td>
</tr>
<tr>
<td>1</td>
<td>18      Hawaiian Is. - Maui</td>
</tr>
<tr>
<td>1</td>
<td>19      Hawaiian Is. - Hawaii</td>
</tr>
<tr>
<td>1</td>
<td>20      Alaska - Kodiak Is.</td>
</tr>
<tr>
<td>1</td>
<td>21      Antarctica - Guest Is.</td>
</tr>
<tr>
<td>1</td>
<td>22      Tahiti</td>
</tr>
<tr>
<td>1</td>
<td>23      Canada - Queen Charlotte Is.</td>
</tr>
<tr>
<td>1</td>
<td>24      Canada - Vancouver Is.</td>
</tr>
<tr>
<td>1</td>
<td>25      Canada - Banks Is.</td>
</tr>
<tr>
<td>1</td>
<td>26      Canada - Great Bear Lake</td>
</tr>
<tr>
<td>1</td>
<td>27      Canada - Prince Patrick Is.</td>
</tr>
<tr>
<td>1</td>
<td>28      Canada - Eglinton Is.</td>
</tr>
<tr>
<td>1</td>
<td>29      Canada - Victoria Is.</td>
</tr>
<tr>
<td>1</td>
<td>30      Canada - Melville Is.</td>
</tr>
<tr>
<td>1</td>
<td>31      Canada - Great Slave Lake</td>
</tr>
<tr>
<td>1</td>
<td>32      Canada - Brock Is.</td>
</tr>
<tr>
<td>1</td>
<td>33      Canada - Borden Is.</td>
</tr>
<tr>
<td>1</td>
<td>34      Canada - Mackenzie King Is.</td>
</tr>
<tr>
<td>1</td>
<td>35      Canada - Lake Athabasca</td>
</tr>
<tr>
<td>1</td>
<td>36      Antarctica - Bear Is.</td>
</tr>
<tr>
<td>1</td>
<td>37      Canada - Ellef Ringnes and Amund Ringnes Is.</td>
</tr>
<tr>
<td>1</td>
<td>38      Canada - Lougheed Is.</td>
</tr>
<tr>
<td>1</td>
<td>39      Canada - Bathurst Is.</td>
</tr>
<tr>
<td>1</td>
<td>40      Canada - Byam Martin Is.</td>
</tr>
<tr>
<td>1</td>
<td>41      Canada - Reindeer Lake</td>
</tr>
<tr>
<td>1</td>
<td>42      Canada - Prince of Wales Is.</td>
</tr>
<tr>
<td>1</td>
<td>43      Canada - Lake Winnipegosis</td>
</tr>
<tr>
<td>1</td>
<td>44      Canada - Lake Manitoba</td>
</tr>
<tr>
<td>1</td>
<td>45      Canada - Meighen Is.</td>
</tr>
<tr>
<td>1</td>
<td>46      Canada - Lake Winnipeg</td>
</tr>
<tr>
<td>1</td>
<td>47      Canada - Cornwall Is.</td>
</tr>
<tr>
<td>1</td>
<td>48      Canada - Cornwallis Is.</td>
</tr>
<tr>
<td>1</td>
<td>49      Canada - Devon Is.</td>
</tr>
<tr>
<td>1</td>
<td>50      Canada - Axel Heiberg Is.</td>
</tr>
<tr>
<td>1</td>
<td>51      Lake Superior</td>
</tr>
<tr>
<td>1</td>
<td>52      Galapagos Is.</td>
</tr>
<tr>
<td>1</td>
<td>53      Canada - Ellesmere Is.</td>
</tr>
<tr>
<td>1</td>
<td>54      Canada - Baffin Is.</td>
</tr>
<tr>
<td>1</td>
<td>55      Lakes Michigan and Huron</td>
</tr>
<tr>
<td>1</td>
<td>56      Canada - Southampton Is.</td>
</tr>
<tr>
<td>Page</td>
<td>Region</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Cuba</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Coats Is.</td>
</tr>
<tr>
<td>1</td>
<td>Lake Erie</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Manitoulin Is.</td>
</tr>
<tr>
<td>1</td>
<td>Cuba - Isla de Pinos</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Akimiski Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Bylot Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Mansel Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Belcher Is.</td>
</tr>
<tr>
<td>1</td>
<td>Lake Ontario</td>
</tr>
<tr>
<td>1</td>
<td>Bahamas</td>
</tr>
<tr>
<td>1</td>
<td>Cuba - Isla de Chilo</td>
</tr>
<tr>
<td>1</td>
<td>Long Is.</td>
</tr>
<tr>
<td>1</td>
<td>Turks and Caicos Is.</td>
</tr>
<tr>
<td>1</td>
<td>Green</td>
</tr>
<tr>
<td>1</td>
<td>Curacao</td>
</tr>
<tr>
<td>1</td>
<td>Puerto Rico</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Anticosti Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Anvers Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Prince Edward Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Brabant Is.</td>
</tr>
<tr>
<td>1</td>
<td>Trinidad and Tobago</td>
</tr>
<tr>
<td>1</td>
<td>Guadeloupe - west</td>
</tr>
<tr>
<td>1</td>
<td>Guadeloupe - east</td>
</tr>
<tr>
<td>1</td>
<td>Faulkland Is.</td>
</tr>
<tr>
<td>1</td>
<td>Martinique</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Livingston Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - King George Is.</td>
</tr>
<tr>
<td>1</td>
<td>Newfoundland</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Joinville Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Coronation Is.</td>
</tr>
<tr>
<td>1</td>
<td>South Georgia Is.</td>
</tr>
<tr>
<td>1</td>
<td>Iceland</td>
</tr>
<tr>
<td>1</td>
<td>Greenland - off-shore island</td>
</tr>
<tr>
<td>1</td>
<td>Africa/Eurasia</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Caribbean</td>
</tr>
<tr>
<td>1</td>
<td>Cuba</td>
</tr>
<tr>
<td>1</td>
<td>Cuba - Isla de Pinos</td>
</tr>
<tr>
<td>1</td>
<td>Cuba - Isla de Chilo</td>
</tr>
<tr>
<td>1</td>
<td>Long Is.</td>
</tr>
<tr>
<td>1</td>
<td>Turks and Caicos Is.</td>
</tr>
<tr>
<td>1</td>
<td>Green</td>
</tr>
<tr>
<td>1</td>
<td>Curacao</td>
</tr>
<tr>
<td>1</td>
<td>Puerto Rico</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Anticosti Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Anvers Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canada - Prince Edward Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Brabant Is.</td>
</tr>
<tr>
<td>1</td>
<td>Trinidad and Tobago</td>
</tr>
<tr>
<td>1</td>
<td>Guadeloupe - west</td>
</tr>
<tr>
<td>1</td>
<td>Guadeloupe - east</td>
</tr>
<tr>
<td>1</td>
<td>Faulkland Is.</td>
</tr>
<tr>
<td>1</td>
<td>Martinique</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Livingston Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - King George Is.</td>
</tr>
<tr>
<td>1</td>
<td>Newfoundland</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Joinville Is.</td>
</tr>
<tr>
<td>1</td>
<td>Antarctica - Coronation Is.</td>
</tr>
<tr>
<td>1</td>
<td>South Georgia Is.</td>
</tr>
<tr>
<td>1</td>
<td>Iceland</td>
</tr>
<tr>
<td>1</td>
<td>Greenland - off-shore island</td>
</tr>
<tr>
<td>1</td>
<td>Africa/Eurasia</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Canary Is.</td>
</tr>
<tr>
<td>1</td>
<td>Caribbean</td>
</tr>
</tbody>
</table>

August 1987
<table>
<thead>
<tr>
<th>Number</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 233 2</td>
<td>Washington - island</td>
</tr>
<tr>
<td>2 234 4</td>
<td>California - San Miguel Is.</td>
</tr>
<tr>
<td>2 235 4</td>
<td>California - Santa Rosa Is.</td>
</tr>
<tr>
<td>2 236 1</td>
<td>Lake Tahoe</td>
</tr>
<tr>
<td>2 237 5</td>
<td>Nevada</td>
</tr>
<tr>
<td>2 238 4</td>
<td>California - Santa Cruz Is.</td>
</tr>
<tr>
<td>2 239 4</td>
<td>California - San Nicholas Is.</td>
</tr>
<tr>
<td>2 240 4</td>
<td>California - San Clemente Is.</td>
</tr>
<tr>
<td>2 241 4</td>
<td>California - Santa Catalina Is.</td>
</tr>
<tr>
<td>2 242 6</td>
<td>Idaho</td>
</tr>
<tr>
<td>2 243 5</td>
<td>Montana</td>
</tr>
<tr>
<td>2 244 1</td>
<td>Lake Mead (west)</td>
</tr>
<tr>
<td>2 245 2</td>
<td>Arizona</td>
</tr>
<tr>
<td>2 246 1</td>
<td>Lake Mead (east)</td>
</tr>
<tr>
<td>2 247 3</td>
<td>Utah</td>
</tr>
<tr>
<td>2 248 1</td>
<td>Great Salt Lake</td>
</tr>
<tr>
<td>2 249 4</td>
<td>Wyoming</td>
</tr>
<tr>
<td>2 250 5</td>
<td>New Mexico</td>
</tr>
<tr>
<td>2 251 6</td>
<td>Colorado</td>
</tr>
<tr>
<td>2 252 2</td>
<td>Texas</td>
</tr>
<tr>
<td>2 253 5</td>
<td>Nebraska</td>
</tr>
<tr>
<td>2 254 2</td>
<td>South Dakota</td>
</tr>
<tr>
<td>2 255 3</td>
<td>North Dakota</td>
</tr>
<tr>
<td>2 256 3</td>
<td>Oklahoma</td>
</tr>
<tr>
<td>2 257 4</td>
<td>Kansas</td>
</tr>
<tr>
<td>2 258 2</td>
<td>Texas - island</td>
</tr>
<tr>
<td>2 259 2</td>
<td>Texas - Padre Is.</td>
</tr>
<tr>
<td>2 260 4</td>
<td>Minnesota</td>
</tr>
<tr>
<td>2 261 1</td>
<td>Lake Texoma</td>
</tr>
<tr>
<td>2 262 6</td>
<td>Iowa</td>
</tr>
<tr>
<td>2 263 2</td>
<td>Missouri</td>
</tr>
<tr>
<td>2 264 1</td>
<td>Minnesota - Lake of the Woods</td>
</tr>
<tr>
<td>2 265 1</td>
<td>Minnesota - Shoal Lake</td>
</tr>
<tr>
<td>2 266 4</td>
<td>Minnesota - Lake of the Woods - land west of lake</td>
</tr>
<tr>
<td>2 267 2</td>
<td>Texas - island</td>
</tr>
<tr>
<td>2 268 4</td>
<td>Minnesota - Lake of the Woods - island</td>
</tr>
<tr>
<td>2 269 4</td>
<td>Minnesota - Lake of the Woods - island</td>
</tr>
<tr>
<td>2 270 5</td>
<td>Arkansas</td>
</tr>
<tr>
<td>2 271 4</td>
<td>Minnesota - Lake of the Woods - island</td>
</tr>
<tr>
<td>2 272 4</td>
<td>Louisiana</td>
</tr>
<tr>
<td>2 273 1</td>
<td>Louisiana - Sabine Lake</td>
</tr>
<tr>
<td>2 274 1</td>
<td>Minnesota - Rainy Lake</td>
</tr>
<tr>
<td>2 275 1</td>
<td>Louisiana - Calcasieu Lake</td>
</tr>
<tr>
<td>2 276 1</td>
<td>Minnesota - Namakan Lake</td>
</tr>
<tr>
<td>2 277 3</td>
<td>Wisconsin</td>
</tr>
<tr>
<td>2 278 1</td>
<td>Louisiana - Grand Lake</td>
</tr>
<tr>
<td>2 279 4</td>
<td>Minnesota - little piece</td>
</tr>
<tr>
<td>2 280 1</td>
<td>Louisiana - White Lake</td>
</tr>
<tr>
<td>2 281 1</td>
<td>Lake Superior</td>
</tr>
<tr>
<td>2 282 4</td>
<td>Louisiana - Marsh Is.</td>
</tr>
<tr>
<td>2 283 2</td>
<td>Mississippi</td>
</tr>
<tr>
<td>2 284 4</td>
<td>Illinois</td>
</tr>
<tr>
<td>2 285 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 286 2</td>
<td>Lake Superior - Madeline Is.</td>
</tr>
<tr>
<td>2 287 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 288 2</td>
<td>Lake Superior - Oak Is.</td>
</tr>
<tr>
<td>2 289 2</td>
<td>Lake Superior - Stockton Is.</td>
</tr>
<tr>
<td>2 290 1</td>
<td>Louisiana - Lake Maurepas</td>
</tr>
<tr>
<td>2 291 4</td>
<td>Louisiana - Timbalier Is.</td>
</tr>
<tr>
<td>2 292 2</td>
<td>Lake Superior - Outer Is.</td>
</tr>
<tr>
<td>2 293 2</td>
<td>Michigan (upper)</td>
</tr>
<tr>
<td>2 294 1</td>
<td>Louisiana - Lake Pontchartrain</td>
</tr>
<tr>
<td>2 295 3</td>
<td>Tennessee</td>
</tr>
<tr>
<td>2 296 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 297 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 298 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 299 6</td>
<td>Kentucky</td>
</tr>
<tr>
<td>2 300 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 301 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 302 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 303 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 304 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 305 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 306 4</td>
<td>Louisiana - island</td>
</tr>
<tr>
<td>2 307 2</td>
<td>Lake Superior - Isle Royale</td>
</tr>
<tr>
<td>2 308 2</td>
<td>Lake Superior - Pie Is.</td>
</tr>
<tr>
<td>2 309 4</td>
<td>Louisiana - Cat Is.</td>
</tr>
<tr>
<td>2 310 5</td>
<td>Alabama</td>
</tr>
<tr>
<td>2 311 5</td>
<td>Indiana</td>
</tr>
<tr>
<td>2 312 1</td>
<td>Lakes Michigan and Huron</td>
</tr>
<tr>
<td>2 313 4</td>
<td>Florida</td>
</tr>
<tr>
<td>2 314 2</td>
<td>Lake Michigan - Washington Is.</td>
</tr>
<tr>
<td>2 315 2</td>
<td>Michigan (lower)</td>
</tr>
<tr>
<td>2 316 2</td>
<td>Lake Superior - Grand Is.</td>
</tr>
<tr>
<td>2 317 2</td>
<td>Lake Michigan - Manitou Is.</td>
</tr>
<tr>
<td>2 318 2</td>
<td>Lake Superior - Michipicoten Is.</td>
</tr>
<tr>
<td>2 319 1</td>
<td>Florida - lake</td>
</tr>
</tbody>
</table>

August 1987
<table>
<thead>
<tr>
<th>Page</th>
<th>Line</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>413</td>
<td>Lake Champlain - Grand Isle</td>
</tr>
<tr>
<td>2</td>
<td>414</td>
<td>New Hampshire - lake</td>
</tr>
<tr>
<td>2</td>
<td>415</td>
<td>Canada - lake</td>
</tr>
<tr>
<td>2</td>
<td>416</td>
<td>New Hampshire - lake</td>
</tr>
<tr>
<td>2</td>
<td>417</td>
<td>Long Is. - Great South Bay</td>
</tr>
<tr>
<td>2</td>
<td>418</td>
<td>New Hampshire</td>
</tr>
<tr>
<td>2</td>
<td>419</td>
<td>New York - Shelter Is.</td>
</tr>
<tr>
<td>2</td>
<td>420</td>
<td>New York - Fishers Is.</td>
</tr>
<tr>
<td>2</td>
<td>421</td>
<td>Rhode Is.</td>
</tr>
<tr>
<td>2</td>
<td>422</td>
<td>Rhode Is. - Block Is.</td>
</tr>
<tr>
<td>2</td>
<td>423</td>
<td>Rhode Is. - island</td>
</tr>
<tr>
<td>2</td>
<td>424</td>
<td>Rhode Is. - island</td>
</tr>
<tr>
<td>2</td>
<td>425</td>
<td>Rhode Is. - island</td>
</tr>
<tr>
<td>2</td>
<td>426</td>
<td>Massachusetts - bay</td>
</tr>
<tr>
<td>2</td>
<td>427</td>
<td>Maine</td>
</tr>
<tr>
<td>2</td>
<td>428</td>
<td>Massachusetts - Martha's Vineyard</td>
</tr>
<tr>
<td>2</td>
<td>429</td>
<td>Massachusetts - Nantucket Is.</td>
</tr>
<tr>
<td>2</td>
<td>430</td>
<td>Massachusetts - inlet</td>
</tr>
<tr>
<td>2</td>
<td>431</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>432</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>433</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>434</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>435</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>436</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>437</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>438</td>
<td>Maine - island</td>
</tr>
<tr>
<td>2</td>
<td>439</td>
<td>Maine - island</td>
</tr>
<tr>
<td>3</td>
<td>440</td>
<td>7 Antarctica</td>
</tr>
<tr>
<td>3</td>
<td>441</td>
<td>1 Oceans</td>
</tr>
<tr>
<td>3</td>
<td>442</td>
<td>6 USSR</td>
</tr>
<tr>
<td>3</td>
<td>443</td>
<td>6 USSR - Wrangel Is.</td>
</tr>
<tr>
<td>3</td>
<td>444</td>
<td>3 Alaska - Atka Is.</td>
</tr>
<tr>
<td>3</td>
<td>445</td>
<td>6 New Zealand - Chatham Is.</td>
</tr>
<tr>
<td>3</td>
<td>446</td>
<td>3 Samoa - Savaii</td>
</tr>
<tr>
<td>3</td>
<td>447</td>
<td>3 Samoa - Upolu</td>
</tr>
<tr>
<td>3</td>
<td>448</td>
<td>3 Alaska - Saint Lawrence Is.</td>
</tr>
<tr>
<td>3</td>
<td>449</td>
<td>3 Alaska - Umnak Is.</td>
</tr>
<tr>
<td>3</td>
<td>450</td>
<td>3 Alaska</td>
</tr>
<tr>
<td>3</td>
<td>451</td>
<td>3 Alaska - Unalaska Is.</td>
</tr>
<tr>
<td>3</td>
<td>452</td>
<td>3 Alaska - Nunivak Is.</td>
</tr>
<tr>
<td>3</td>
<td>453</td>
<td>7 Antarctica - Roosevelt Is.</td>
</tr>
<tr>
<td>3</td>
<td>454</td>
<td>3 Hawaiian Is. - Niihau</td>
</tr>
<tr>
<td>3</td>
<td>455</td>
<td>3 Hawaiian Is. - Kauai</td>
</tr>
<tr>
<td>3</td>
<td>456</td>
<td>3 Hawaiian Is. - Oahu</td>
</tr>
<tr>
<td>3</td>
<td>457</td>
<td>3 Hawaiian Is. - Maui</td>
</tr>
<tr>
<td>3</td>
<td>458</td>
<td>3 Hawaiian Is. - Hawaii</td>
</tr>
<tr>
<td>3</td>
<td>459</td>
<td>3 Alaska - Kodiak Is.</td>
</tr>
<tr>
<td>3</td>
<td>460</td>
<td>7 Antarctica - Guest Is.</td>
</tr>
<tr>
<td>3</td>
<td>461</td>
<td>3 Tahiti</td>
</tr>
<tr>
<td>3</td>
<td>462</td>
<td>6 Canada</td>
</tr>
<tr>
<td>3</td>
<td>463</td>
<td>6 Canada - Queen Charlotte Is.</td>
</tr>
<tr>
<td>3</td>
<td>464</td>
<td>6 Canada - Vancouver Is.</td>
</tr>
<tr>
<td>3</td>
<td>465</td>
<td>6 Canada - Banks Is.</td>
</tr>
<tr>
<td>3</td>
<td>466</td>
<td>1 Canada - Great Bear Lake</td>
</tr>
<tr>
<td>3</td>
<td>467</td>
<td>2 US - Washington</td>
</tr>
<tr>
<td>3</td>
<td>468</td>
<td>3 US - Oregon</td>
</tr>
<tr>
<td>3</td>
<td>469</td>
<td>4 US - California</td>
</tr>
<tr>
<td>3</td>
<td>470</td>
<td>6 Canada - Prince Patrick Is.</td>
</tr>
<tr>
<td>3</td>
<td>471</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>472</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>473</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>474</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>475</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>476</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>477</td>
<td>2 US - Washington - island</td>
</tr>
<tr>
<td>3</td>
<td>478</td>
<td>4 US - California - San Miguel Is.</td>
</tr>
<tr>
<td>3</td>
<td>479</td>
<td>4 US - California - Santa Rosa Is.</td>
</tr>
<tr>
<td>3</td>
<td>480</td>
<td>1 US - Lake Tahoe</td>
</tr>
<tr>
<td>3</td>
<td>481</td>
<td>5 US - Nevada</td>
</tr>
<tr>
<td>3</td>
<td>482</td>
<td>4 US - California - Santa Cruz Is.</td>
</tr>
<tr>
<td>3</td>
<td>483</td>
<td>4 US - California - San Nicholas Is.</td>
</tr>
<tr>
<td>3</td>
<td>484</td>
<td>6 Canada - Eglinton Is.</td>
</tr>
<tr>
<td>3</td>
<td>485</td>
<td>6 Canada - Victoria Is.</td>
</tr>
<tr>
<td>3</td>
<td>486</td>
<td>4 US - California - San Clemente Is.</td>
</tr>
<tr>
<td>3</td>
<td>487</td>
<td>4 US - California - Santa Catalina Is.</td>
</tr>
<tr>
<td>3</td>
<td>488</td>
<td>6 Canada - Melville Is.</td>
</tr>
<tr>
<td>3</td>
<td>489</td>
<td>6 US - Idaho</td>
</tr>
<tr>
<td>3</td>
<td>490</td>
<td>1 Canada - Great Slave Lake</td>
</tr>
<tr>
<td>3</td>
<td>491</td>
<td>6 Mexico</td>
</tr>
<tr>
<td>3</td>
<td>492</td>
<td>5 US - Montana</td>
</tr>
<tr>
<td>3</td>
<td>493</td>
<td>6 Canada - Brock Is.</td>
</tr>
<tr>
<td>3</td>
<td>494</td>
<td>1 US - Lake Mead (west)</td>
</tr>
<tr>
<td>3</td>
<td>495</td>
<td>2 US - Arizona</td>
</tr>
<tr>
<td>3</td>
<td>496</td>
<td>1 US - Lake Mead (east)</td>
</tr>
<tr>
<td>3</td>
<td>497</td>
<td>3 US - Utah</td>
</tr>
<tr>
<td>3</td>
<td>498</td>
<td>6 Canada - Borden Is.</td>
</tr>
<tr>
<td>3</td>
<td>499</td>
<td>6 Canada - Mackenzie King Is.</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>1 US - Great Salt Lake</td>
</tr>
</tbody>
</table>
3 582 2 Belize
3 583 2 US - Lake Superior - Pie Is.
3 584 4 US - Louisiana - Cat Is.
3 585 5 US - Alabama
3 586 5 US - Indiana
3 587 1 US - Lakes Michigan and Huron
3 588 6 Nicaragua
3 589 4 US - Florida
3 590 6 Canada - Southampton Is.
3 592 2 US - Michigan (lower)
3 596 2 Costa Rica
3 597 1 US - Florida - lake
3 598 6 US - Georgia
3 600 4 US - Florida - island
3 601 1 US - Lake Seminole
3 602 4 Cuba
3 603 4 US - Ohio
3 605 2 US - Lake Superior - island
3 606 1 US - Sault Ste. Marie
3 607 4 US - North Carolina
3 608 1 US - Lake George
3 609 2 US - Lake Huron - Drummond Is.
3 610 6 Canada - Coats Is.
3 611 2 US - Virginia
3 612 2 US - Lake Huron - Cockburn Is.
3 613 1 US - Lake Erie
3 614 2 US - South Carolina
3 616 4 Cuba - Isla de Pinos
3 617 5 Panama
3 618 1 US - Lake St. Clair
3 620 3 US - West Virginia
3 621 1 US - Clark Hill Reservoir
3 622 4 US - Florida - Sanibel Is.
3 624 6 Canada - Akimiski Is.
3 625 2 US - Lake Huron - Fitzwilliam Is.
3 626 4 US - Florida - Keys
3 627 4 US - Florida - Keys
3 628 4 US - Florida - Keys
3 629 4 US - Florida - Keys
3 630 5 Peru
3 631 1 US - Florida - Lake Okeechobee
3 632 3 Ecuador
3 633 4 US - Florida - island
3 634 6 Canada - Bylot Is.
3 635 4 US - Florida - island
3 636 4 US - Florida - island
3 637 4 US - Florida - island
3 638 6 US - Pennsylvania
3 639 4 US - Florida - island
3 640 2 US - Lake Erie - Long Point (land)
3 642 6 Canada - Mansel Is.
3 643 6 Canada - Belcher Is.
3 644 3 US - New York
3 645 1 US - Lake Ontario
3 646 4 US - Maryland
3 647 2 Colombia
3 648 4 US - Bahamas - Grand Bahama
3 649 4 US - Bahamas - Andros
3 650 5 Jamaica
3 651 4 US - North Carolina - Cape Fear
3 652 4 US - Bahamas - Andros
3 653 4 US - Bahamas - Andros
3 654 4 US - Bahamas - Great Abaco and Little Abaco
3 655 4 US - Bahamas - Andros
3 656 4 US - Bahamas - Grand Bahama
3 657 4 US - Bahamas - Grand Bahama
3 658 4 US - Bahamas - Andros
3 661 6 Canada - Prince Charles Is.
<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>US - Lake Ontario - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Bahamas - Eleuthera Is.</td>
</tr>
<tr>
<td>3</td>
<td>US - Bahamas - Eleuthera Is.</td>
</tr>
<tr>
<td>3</td>
<td>Antarctica - Alexander Is.</td>
</tr>
<tr>
<td>3</td>
<td>US - Lake Ontario - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Maryland - island</td>
</tr>
<tr>
<td>3</td>
<td>Antarctica - Charcot Is.</td>
</tr>
<tr>
<td>3</td>
<td>US - Lake Ontario - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Maryland - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Virginia - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Maryland - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Virginia</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Maryland - inlet</td>
</tr>
<tr>
<td>3</td>
<td>US - Virginia - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - Delaware</td>
</tr>
<tr>
<td>3</td>
<td>US - Bahamas - Cat Is.</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
<tr>
<td>3</td>
<td>US - North Carolina - island</td>
</tr>
</tbody>
</table>

3 710 3 Venezuela
3 711 1 US - Lake Champlain
3 712 2 US - Vermont
3 713 3 Argentina
3 714 2 US - Lake Champlain - Grand Isle
3 715 1 US - New Hampshire - lake
3 716 1 Canada - lake
3 717 1 US - New Hampshire - lake
3 718 2 Greenland
3 719 1 US - Long Is. - Great South Bay
3 720 3 US - New Hampshire
3 723 6 US - Rhode Is.
3 724 3 Dominican Republic
3 725 6 US - Rhode Is. - Block Is.
3 726 6 US - Rhode Is. - island
3 727 6 US - Rhode Is. - island
3 728 6 US - Rhode Is. - island
3 729 1 US - Massachusetts - bay
3 730 5 US - Maine
3 731 4 US - Massachusetts - Marthas Vineyard
3 733 1 US - Massachusetts - inlet
3 734 4 Bolivia
3 735 3 Curacao
3 736 5 US - Maine - island
3 737 5 US - Maine - island
3 738 5 US - Maine - island
3 739 3 Argentina - Tierra del Fuego
3 740 5 US - Maine - island
3 741 5 US - Maine - island
3 742 5 US - Maine - island
3 743 5 US - Maine - island
3 744 5 US - Maine - island
3 745 5 US - Maine - island
3 746 3 Puerto Rico
3 747 6 Canada - Anticosti Is.
3 748 7 Antarctica - Anvers Is.
3 749 6 Canada - Prince Edward Is.
3 750 7 Antarctica - Brabant Is.
3 751 5 Paraguay
3 752 6 Trinidad and Tobago

August 1987
| 3 753 | Guadeloupe - west |
| 3 754 | Guadeloupe - east |
| 3 755 | Guyana |
| 3 756 | Falkland Is. |
| 3 757 | Martinique |
| 3 758 | Antarctica - Livingston Is. |
| 3 759 | Antarctica - King George Is. |
| 3 760 | Canada - Newfoundland |
| 3 761 | Uruguay |
| 3 762 | Surinam |
| 3 763 | Antarctica - Joinville Is. |
| 3 764 | French Guiana |
| 3 765 | Antarctica - Coronation Is. |
| 3 766 | South Georgia Is. |
| 3 767 | Iceland |
| 3 768 | Greenland - off-shore island |
| 3 769 | Senegal |
| 3 770 | Western Sahara |
| 3 771 | Mauritania |
| 3 772 | Canary Is. |
| 3 774 | Guinea-Bissau |
| 3 775 | Gambia |
| 3 776 | Guinea |
| 3 777 | Canary Is. |
| 3 778 | Canary Is. |
| 3 779 | Sierra Leone |
| 3 780 | Morocco |
| 3 781 | Mali |
| 3 782 | Liberia |
| 3 783 | Ireland |
| 3 784 | Portugal |
| 3 785 | Spain |
| 3 786 | Algeria |
| 3 787 | Ivory Coast |
| 3 788 | Northern Ireland |
| 3 789 | Faeroe Is. |
| 3 790 | Outer Hebrides |
| 3 791 | Scotland |
| 3 792 | England |
| 3 793 | Upper Volta |
| 3 794 | Wales |
| 3 795 | France |
| 3 796 | Ghana |
| 3 797 | Togo |
| 3 798 | Niger |
| 3 799 | Benin |
| 3 800 | Andorra |
| 3 801 | Mallorca |
| 3 802 | Belgium |
| 3 803 | Nigeria |
| 3 804 | Netherlands |
| 3 805 | Norway |
| 3 806 | Luxembourg |
| 3 807 | Switzerland |
| 3 808 | West Germany |
| 3 809 | Italy |
| 3 810 | Tunisia |
| 3 811 | Denmark - Jutland |
| 3 812 | Sardinia |
| 3 813 | Cameroon |
| 3 814 | Macias Nguema Biyogo |
| 3 815 | Corsica |
| 3 816 | Gabon |
| 3 817 | Equatorial Guinea |
| 3 818 | Libya |
| 3 819 | Liechtenstein |
| 3 820 | Austria |
| 3 821 | Denmark - Fyn |
| 3 822 | East Germany |
| 3 823 | Denmark - Lolland |
| 3 824 | Svalbard |
| 3 825 | Denmark - Sjaelland |
| 3 826 | Sweden |
| 3 827 | Congo |
| 3 828 | Namibia |
| 3 829 | Angola |
| 3 830 | Angola - exclae called Cabinda |
| 3 831 | Czechoslovakia |
| 3 832 | Zaire |
| 3 833 | Sicily |
| 3 834 | Lake Chad |
| 3 835 | Yugoslavia |
| 3 836 | Chad |
| 3 837 | Poland |
| 3 838 | Central African Republic |
| 3 839 | Hungary |
| 3 840 | South Africa |
| 3 841 | Gotland |
| 3 842 | Albania |
| 3 843 | Malaysia - Malay Peninsula |
| 3 844 | Greece |
| 3 845 | Botswana |
| 3 846 | Romania |

August 1987
3 847 3 Finland
3 848 2 Greece
3 849 6 USSR - Ostrov Saaremaa
3 850 3 Sudan

3 851 5 Zambia
3 852 6 USSR - Ostrov Hiiumaa
3 853 5 Bulgaria
3 854 2 Crete
3 855 6 Egypt
3 856 3 Zimbabwe
3 857 3 Turkey
3 858 6 Lesotho
3 859 1 Black Sea
3 860 2 Rhodes

3 861 5 Rwanda
3 862 1 Lake Tanganyika
3 863 4 Burundi
3 864 2 Uganda
3 865 3 Tanzania
3 866 1 USSR - Ozero Ladozhskoye (Lake Ladoga)
3 867 4 Mozambique
3 868 6 Swaziland
3 869 1 Lake Victoria
3 870 3 Cyprus

3 871 2 Malawi
3 872 4 Ethiopia
3 873 5 Kenya
3 874 1 Lake Nyasa
3 875 2 Israel
3 876 1 USSR - Azov Sea
3 877 1 USSR - Onazhskoye Ozero
3 878 3 Saudi Arabia
3 879 5 Jordan
3 880 3 Lebanon

3 881 4 Syria
3 882 6 Iraq
3 883 2 Somalia
3 884 3 Djibouti
3 885 5 Yemen (Sana)
3 886 3 Madagascar
3 887 6 Yemen (Aden)
3 888 5 Iran
3 889 6 USSR - Franz Josef Land
3 890 4 Unnamed neutral zone between Iraq and Saudi Arabia
3 891 1 Caspian Sea

3 892 6 USSR - Ostrov Kolguyev
3 893 2 United Arab Emirates
3 894 4 Qatar
3 895 6 USSR - Novaya Zemlya
3 896 4 Oman - northern section
3 897 6 USSR - Franz Josef Land
3 898 6 Yemen (Aden) - Socotra (island)
3 899 6 USSR - Franz Josef Land
3 900 3 Reunion Is.

3 901 6 USSR - Franz Josef Land
3 902 4 Oman - southern section
3 903 3 Mauritius
3 904 1 USSR - Aral Sea
3 905 6 USSR - Franz Josef Land
3 906 3 Afghanistan
3 907 4 Pakistan
3 908 6 USSR - Franz Josef Land
3 909 2 India
3 910 3 Ille Kerguelen

3 911 1 USSR - Lake Balkhash (west)
3 912 5 China
3 913 1 USSR - Lake Balkhash (east)
3 914 4 Disputed area between India and China
3 915 5 Sri Lanka (Ceylon)
3 916 3 Nepal
3 917 3 Mongolia
3 918 4 Bangladesh
3 919 3 Bhutan
3 920 6 USSR - Ostrov Oktyabr'skoy Revolyutsii, Komsomolets, Pioneer

3 921 3 Burma
3 922 5 Indonesia - Sumatra
3 923 5 Indonesia - Nias
3 924 5 Thailand
3 925 5 Indonesia - Siberut
3 926 6 USSR - Ostrov Bol'shevik
3 927 6 Laos
3 928 2 Cambodia
3 929 4 Vietnam
3 930 1 USSR - Lake Baikal

3 931 5 Indonesia - Bangka
3 932 5 Indonesia - Java
3 933 5 Indonesia - Belitung
3 934 5 China - Hainan Dao (island)
3 935 5 Indonesia - Borneo - Kalimantan
3 936 6 Malaysia - Borneo - Malaysia Timor
3 937 6 USSR - Ostrov Bol’shoy Begichev
3 938 5 Indonesia - Bawean
3 939 3 Australia
3 940 3 Borneo - Brunei
3 941 5 Indonesia - Bali
3 942 5 Indonesia - Sumbawa & Lombok
3 943 4 Philippines - Palawan
3 944 5 Indonesia - Sulawesi (Celebes)
3 945 5 Indonesia - Sumba
3 946 5 Indonesia - Flores
3 947 4 Philippines - Luzon
3 948 2 Taiwan
3 949 4 Philippines - Mindoro
3 950 4 Philippines - Panay, Negros, and Bohol
3 951 4 Philippines - Mindanao
3 952 5 Indonesia - island
3 953 5 Indonesia - Timor
3 954 5 Indonesia - Alor
3 955 4 Philippines - Samar
3 956 2 North Korea
3 957 5 Indonesia - Wetar
3 958 5 Indonesia - Buru
3 959 4 South Korea
3 960 5 Indonesia - Obi
3 961 5 Indonesia - Halmahera
3 962 5 Indonesia - Ceram
3 963 3 Japan - Kyushu
3 964 3 Australia - Melville and Bathurst Is.
3 965 5 Indonesia - Waigeo
3 966 5 Indonesia - New Guinea - Irian Jaya
3 967 3 Japan - Honshu
3 968 5 Indonesia - Kepulauan Tanimbar
3 969 3 Japan - Shikoku
3 970 5 Indonesia - Kepulauan Aru
3 971 5 Indonesia - Biak and Kepulauan Schouten
3 972 3 Australia - Kangaroo Is.
3 973 6 USSR - Ostrov Kotel’nnyy and Ostrov Faddeyevskiy
3 974 6 USSR - Shantarskiye Ostrova
3 975 3 Japan - Hokkaido
3 976 6 USSR - Ostrov Malyy Lyakhovskiy
3 977 6 USSR - Ostrov Bol’shoy Lyakhovskiy
3 978 3 New Guinea - Papua
3 979 6 USSR - Ostrov Sakhalin
3 980 3 Australia - Tasmania
3 981 6 USSR - Ostrov Novaya Sibir’
3 982 6 USSR - Ostrov Iturup
3 983 3 Bismarck Archipelago - New Britain
3 984 6 USSR - Ostrov Urup
3 985 3 Bismarck Archipelago - New Hanover
3 986 3 Bismarck Archipelago - New Ireland
3 987 3 Solomon Is. - Bougainville
3 988 6 USSR - Ostrov Paramushir
3 989 3 Solomon Is. - Guadalcanal
3 990 3 Solomon Is. - San Cristobal
3 991 6 USSR - Ostrov Karaginskiy
3 992 3 New Caledonia (island)
3 993 7 Antarctica - Ross Is.
3 994 6 New Zealand - South Is.
3 995 3 New Hebrides - Espiritu Santo
3 996 3 New Hebrides - Malekula
3 997 6 New Zealand - Stewart Is.
3 998 3 Alaska - Attu Is.
3 999 6 New Zealand - North Is.
3 1000 6 New Zealand - North Is.
3 1001 3 Fiji - Viti Levu
3 1002 3 Fiji - Vanua Levu
3 1003 2 Kuwait
4 1004 7 Antarctica
4 1005 1 Oceans
4 1006 6 USSR
4 1007 6 USSR - Wrangel Is.
4 1008 3 Alaska - Atka Is.
4 1009 6 New Zealand - Chatham Is.
4 1010 3 Samoa - Savaii
4 1011 3 Samoa - Upolu
4 1012 3 Alaska - Saint Lawrence Is.
4 1013 3 Alaska - Umnak Is.
4 1014 3 Alaska
4 1015 3 Alaska - Unalaska Is.
4 1016 3 Alaska - Nunivak Is.
4 1017 7 Antarctica - Roosevelt Is.

August 1987
4 1111 3 Guadeloupe - west
4 1112 3 Guadeloupe - east
4 1113 4 Guyana
4 1114 5 Falkland Is.
4 1115 3 Martinique
4 1116 7 Antarctica - Livingston Is.
4 1117 7 Antarctica - King George Is.
4 1118 6 Canada - Newfoundland
4 1119 2 Uruguay
4 1120 5 Surinam
4 1121 7 Antarctica - Joinville Is.
4 1122 3 French Guiana
4 1123 7 Antarctica - Coronation Is.
4 1124 5 South Georgia Is.
4 1125 7 Iceland
4 1126 2 Greenland - off-shore island
4 1127 3 Senegal
4 1128 6 Western Sahara
4 1129 4 Mauritania
4 1130 2 Canary Is.
4 1131 4 Guinea-Bissau
4 1132 2 Gambia
4 1133 2 Canary Is.
4 1134 5 Guinea
4 1135 2 Canary Is.
4 1136 2 Canary Is.
4 1137 6 Sierra Leone
4 1138 5 Morocco
4 1139 6 Mali
4 1140 3 Liberia
4 1141 4 Ireland
4 1142 4 Portugal
3 1143 2 Spain
4 1144 2 Austria
4 1145 4 Ivory Coast
4 1146 5 Northern Ireland
4 1147 2 Faeroe Is.
4 1148 2 Outer Hebrides
4 1149 2 Scotland
4 1150 5 England
4 1151 2 Upper Volta
4 1152 3 Wales
4 1153 3 France
4 1154 5 Ghana
4 1155 6 Togo
4 1156 5 Niger
4 1157 3 Benin
4 1158 4 Andorra
4 1159 2 Mallorca
4 1160 6 Belgium
4 1161 4 Nigeria
4 1162 3 Netherlands
4 1163 4 Norway
4 1164 5 Luxembourg
4 1165 5 Switzerland
4 1166 4 West Germany
4 1167 6 Italy
4 1168 5 Tunisia
4 1169 2 Denmark - Jutland
4 1170 6 Sardinia
3 1171 3 Cameroon
4 1172 3 Macias Nguema Biyogo (African island, Fernando Po)
4 1173 3 Corsica
4 1174 5 Gabon
4 1175 6 Equatorial Guinea
4 1176 4 Libya
4 1177 4 Liechtenstein
4 1178 3 Austria
4 1179 2 Denmark - Fyn
4 1180 3 East Germany
4 1181 2 Denmark - Lolland
4 1182 4 Svalbard
4 1183 2 Denmark - Sjaelland
4 1184 5 Sweden
4 1185 2 Congo
4 1186 2 Namibia
4 1187 4 Angola - exclave called Cabinda
4 1188 4 Angola
4 1189 2 Czechoslovakia
4 1190 6 Zaire
4 1191 6 Sicily
4 1192 1 Lake Chad
4 1193 4 Yugoslavia
4 1194 2 Chad
4 1195 5 Poland
4 1196 5 Central African Republic
4 1197 5 Hungary
4 1198 5 South Africa
3 1199 2 Gotland
4 1200 3 Albania
4 1201 6 Malaysia - Malay Peninsula
August 1987
4 1292 5 China - Hainan Dao (island)
4 1293 5 Indonesia - Borneo - Kalimantan
4 1294 6 Malaysia - Borneo - Malaysia Timor
4 1295 6 USSR - Ostrov Bol'shay Begichev
4 1296 5 Indonesia - Bawean
4 1297 3 Australia
4 1298 3 Borneo - Brunei
4 1299 5 Indonesia - Bali
4 1300 5 Indonesia - Sumbawa and Lombok
4 1301 4 Philippines - Palawan
4 1302 5 Indonesia - Sulawesi (Celebes)
4 1303 5 Indonesia - Sumba
4 1304 5 Indonesia - Flores
4 1305 4 Philippines - Luzon
4 1306 2 Taiwan
4 1307 4 Philippines - Mindoro
4 1308 4 Philippines - Panay, Negros, and Bohol
4 1309 4 Philippines - Mindanao
4 1310 5 Indonesia - island
3 1311 5 Indonesia - Timor
4 1312 5 Indonesia - Alor
4 1313 4 Philippines - Samar
4 1314 2 North Korea
4 1315 5 Indonesia - Wetar
4 1316 5 Indonesia - Buru
4 1317 4 South Korea
4 1318 5 Indonesia - Obi
4 1319 5 Indonesia - Halmahera
4 1320 5 Indonesia - Ceram
4 1321 3 Japan - Kyushu
4 1322 3 Australia - Melville and Bathurst Is.
4 1323 5 Indonesia - Waigeo
4 1324 5 Indonesia - New Guinea - Irian Jaya
4 1325 3 Japan - Honshu
4 1326 5 Indonesia - Kepulauan Tanimbar
4 1327 3 Japan - Shikoku
4 1328 5 Indonesia - Kepulauan Schouten
4 1329 5 Indonesia - Biak and Kepulauan Schouten
4 1330 3 Australia - Kangaroo Is.
4 1331 6 USSR - Ostrov Kotel'nyy and Ostrov Faddeyevskiy
4 1332 6 USSR - Shantarskiye Ostrova
4 1333 3 Japan - Hokkaido
4 1334 6 USSR - Ostrov Malyy Lyakhovskiy
4 1335 6 USSR - Ostrov Bol'shay Lyakhovskiy
4 1336 3 New Guinea - Papua
4 1337 6 USSR - Ostrov Sakhalin
4 1338 3 Australia - Tasmania
3 1339 6 USSR - Ostrov Novaya Sibir'
4 1340 6 USSR - Ostrov Iturup
4 1341 3 Bismarck Archipelago - New Britain
4 1342 6 USSR - Ostrov Urup
4 1343 3 Bismarck Archipelago - New Hanover
4 1344 3 Bismarck Archipelago - New Ireland
4 1345 3 Solomon Is. - Bougainville
4 1346 6 USSR - Ostrov Paramushir
4 1347 3 Solomon Is. - Guadalcanal
4 1348 3 Solomon Is. - San Cristobal
4 1349 6 USSR - Ostrov Karaginskiy
4 1350 3 New Caledonia (island)
4 1351 7 Antarctica - Ross Is.
4 1352 6 New Zealand - South Is.
4 1353 3 New Hebrides - Espiritu Santo
4 1354 3 New Hebrides - Malekula
4 1355 6 New Zealand - Stewart Is.
4 1356 3 Alaska - Attu Is.
4 1357 6 New Zealand - North Is.
4 1358 6 New Zealand - North Is.
4 1359 3 Fiji - Viti Levu
4 1360 3 Fiji - Vanua Levu
4 1361 2 Kuwait
EZMAPA
Purpose

EZMAPA allows you to redirect EZMAP output to routines in the package AREAS. Among other things, this makes it possible to create solid-colored world maps.

Usage

To use the routines in the package EZMAPA, you should be familiar with the packages EZMAP and AREAS.

User-level routines in EZMAPA are as follows:

- The subroutine MAPBLA adds boundary lines to an already existing area map. The boundary lines are produced by projecting a selected EZMAP dataset according to a specified EZMAP projection within a specified area. The area map may then be used in various ways (for example, to produce a colored map). (MAPBLA stands for map boundary lines to area map.)

- The subroutine MAPGRM draws lines of longitude and latitude that are masked by the areas defined by the area map. MAPGRM was used to draw the longitude and latitude lines over water and omit them over land masses on the cover of this manual. (MAPGRM stands for map grid, masked.)

- The subroutines MAPITA and MAPIQA add the projections of arbitrary lines, defined by sets of latitude/longitude pairs, to an area map. (MAPITA stands for map it, areas. MAPIQA stands for map it, quit areas.)

- The subroutines MAPITM and MAPIQM draw the projections of arbitrary lines, defined by sets of latitude/longitude pairs, masked by the areas defined by the area map. (MAPITM stands for map it, masked. MAPIQM stands for map it, quit masked.)

- The function MAPACI obtains, for a given area, a color index appropriate for use in generating a colored map that has adjacent areas of different colors. (MAPACI stands for map area color index.)

See "Details of Routines" below for complete descriptions of each routine.

REQUIRED STEPS

You must perform the four steps listed here before calling any of the other EZMAPA routines.

Step One

First, execute the FORTRAN statement

CALL ARINAM (IAM, LAM)

IAM An integer array in which an area map is being constructed.
LAM Length of the array IAM.

("ARINAM" stands for initialize area map.) For more information on ARINAM, see the AREAS documentation.
**Step Two**  
Next, call various EZMAP routines (MAPPOS, MAPROJ, MAPSET, MAPSTC, MAPSTI, and MAPSTR) to define the position of the map on the plotter frame, the projection you want, the EZMAP dataset you are using, and the limits of the map, just as if you were going to call MAPDRW to draw a map.

For more information on these routines, see the EZMAP documentation.

**Step Three**  
Execute the statement

```
CALL MAPINT
```

to initialize EZMAP.

("MAPINT" stands for map initialize.) For more information on MAPINT, see the EZMAP documentation.

**Step Four**  
Execute the statement

```
CALL MAPBLA (IAM)
```

to retrieve, project, and add the selected EZMAP boundary lines to the area map.

IAM  
An integer array in which an area map is being constructed.

("MAPBLA" stands for map boundary lines to area map.) For more information on MAPBLA, see the "Details of Routines" section below.

**POSSIBLE NEXT STEPS**  
After the above steps are completed, you can call routines in the package AREAS to do various things with the area map that has been created. For example, you can call ARSCAM to draw a solid-colored world map. See the "Example" below.

Also, you can now call the other routines in the package EZMAPA:

- MAPGRM, to draw latitude and longitude lines that are masked by the areas defined by the area map.
- MAPITA and MAPIQA, to add arbitrary lines, defined by sets of latitude/longitude pairs, to an area map.
- MAPITM and MAPIQM, to draw arbitrary lines that are masked by the areas defined by an area map. The lines are defined by sets of latitude/longitude pairs.

These routines are described under "Details of Routines" below.

**Entry Points**  
MAPACI MAPBLA MAPGRM MAPIQA MAPIQM MAPITA MAPITM
Common Blocks
- MAPCM1
- MAPCM2
- MAPCM3
- MAPCM4
- MAPCM8
- MAPCMA
- MAPCMB
- MAPCMC
- MAPSAT

Required Routines
All routines in the packages AREAS and EZMAP.

Required GKS
- Level 0A.

I/O
None.

Precision
Single.

Language
FORTRAN 77.

History
EZMAPA was added to the NCAR Graphics package in 1987.

DETAILS OF ROUTINES

The EZMAPA routines are listed here in alphabetical order.

FUNCTION MAPACI

Purpose
MAPACI obtains, for a given area, a color index appropriate for use in creating a color map having adjacent areas of different colors.

Usage
The FORTRAN statement

\[ \text{ICI} = \text{MAPACI}(\text{IAI}) \]

gives ICI the value of the color index (1 through 7, inclusive) corresponding to the area whose area identifier is IAI.

Arguments
- IAI: Area identifier for an area defined by an area map created by MAPBLA.

Important Note
See the "Area Identifiers" section of the EZMAP documentation for the area identifiers associated with various parts of the world map, and for suggested color indices.

SUBROUTINE MAPBLA

Purpose
MAPBLA adds to an area map the boundary lines produced by projecting a selected EZMAP dataset according to a specified EZMAP projection within a specified area.
Usage

CALL MAPBLA (IAM)

One or two groups of boundary lines are added to the area map by a call to MAPBLA. The first group has the group identifier 'G1', with a default value of 1. The group 'G1' consists of a perimeter (either rectangular or elliptical, depending on the value of the EZMAP parameter 'EL') and the set of projected boundary lines implied by your selection of an EZMAP dataset (some combination of continental, U.S., and world political outlines). For certain projections, a limb line may also be included. (The limb is the boundary between a projectable region and an unprojectable one.)

If the EZMAP parameter 'VS' has a value greater than zero, a second group, with group identifier 'G2', is added to the area map. (The default value of 'VS' is 1; the default value of 'G2' is 2.) The purpose of the group 'G2' is to split areas up and thus reduce the number of points required to define a typical area. This is necessary because some hardware devices fail if the number of points defining an area is too large. The group 'G2' consists of a copy of the perimeter and the limb line (if any) plus a set of vertical lines that split the area inside the perimeter into 'VS' vertical strips.

The perimeter and the limb in the groups 'G1' and 'G2' have the following left and right area identifiers:
- 0, for the area inside the perimeter or limb.
- -1, for the area outside the perimeter or limb.

The vertical lines in the group 'G2' have left and right area identifiers of 0.

To set the values of 'G1', 'G2', and 'VS', call the EZMAP routine MAPSTI. To get the current values of 'G1', 'G2', and 'VS', call the EZMAP routine MAPGTI. See the EZMAP documentation for descriptions of the routines MAPSTI and MAPGTI.

Arguments

IAM Integer array in which an area map is being constructed. IAM must previously have appeared in a call to the routine ARINAM.

SUBROUTINE MAPGRM

Purpose MAPGRM draws the same grid of latitude and longitude lines as would be drawn by the EZMAP routine MAPGRD, but the lines are masked by an area map created by MAPBLA.

Usage CALL MAPGRM (IAM, XCS, YCS, MCS, IAI, IAG, MAI, LPR)
Arguments

IAM

Integer array in which an area map has been constructed by MAPBLA.

XCS, YCS

Real arrays, of dimension MCS, to be used by MAPGRM in calls to the user-provided routine LPR.

MCS

Dimension of the arrays XCS and YCS. MCS must be greater than or equal to 2; the value 100 is suggested.

IAI, IAG

Integer arrays, of dimension MAI, to be used by MAPGRM in calls to the user-provided routine LPR.

MAI

Dimension of the arrays IAI and IAG. MAI must be greater than or equal to 2. If other groups of boundary lines are added to the area map (by calling MAPITA and MAPIQA or by calling AREDAM), then MAI must be increased by the number of such groups.

LPR

A line-processing subroutine, which is called once for each piece of the masked grid. Each such piece is contained in exactly one area defined by the area map. LPR must appear in an EXTERNAL statement in the routine that calls MAPGRM. The subroutine LPR must be provided by you and it must have the following structure:

```
SUBROUTINE LPR (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)

(CODE TO PROCESS POLYLINE DEFINED BY XCS, YCS, IAI, AND IAG)

RETURN
END
```

XCS, YCS

Hold the X and Y coordinates, in the fractional coordinate system, of NCS points defining a piece of the grid.

NCS

Number of X and Y coordinates in the arrays XCS and YCS.

IAI, IAG

Hold NAI pairs of identifiers for the area within which the piece of the polyline lies. For each value of I from 1 to NAI, IAI(I) is the area identifier for the area with respect to the group of edges specified by the group identifier IAG(I).

NAI

Number of values in IAI and IAG. NAI will be equal to the number of groups of edges that you put in the area map.

You can assume that MAPGRM has done the following call:

```
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1)
```
This call ensures that, if the normalized device coordinates in XCS and YCS are used in calls to such routines as GPL, the results will be correct. LPR may do its own SET call to achieve some other effect.

SUBROUTINES MAPITA and MAPIQA

**Purpose** MAPBLA uses the routines MAPITA and MAPIQA to insert lines in an area map. You may call them directly to add the projection of an arbitrary line, defined by the latitudes and longitudes of points along it, to the area map.

**Usage**

```plaintext
CALL MAPITA (RLAT,RLON,IFST,IAM,IGP,IDL,IDR)
CALL MAPIQA (IAM,IGP,IDL,IDR)
```

You must call MAPITA once for each point along the line. After your last call to MAPITA for a given line, you must call MAPIQA to signal the end of the line.

The projection of the line segment joining two points on the globe is considered to be the straight-line segment joining the projections of the points; no attempt is made to project it as if it were a portion of a great circle.

- If both endpoints of a line segment are visible, the segment is considered to be entirely visible.
- If both endpoints are invisible, the segment is considered to be entirely invisible.
- If one endpoint is visible and the other is not, a new point is interpolated at the boundary between the visible and invisible portions.

Only the visible portions of the line are added to the area map.

**Note:** Because of the way MAPITA and MAPIQA work, (as described above) points defining a line should not be too far apart on the globe.

**Boundaries:** There are two types of boundaries between visible and invisible regions:

- The *limb* is a boundary between a projectable region and an unprojectable one. The limb may be circular, elliptical, or some other shape, depending on the projection being used. For example, an orthographic projection has as its limb a circle, centered at (0,0), with a radius of 1.
- The *perimeter* is a rectangular or elliptical boundary defined by EZMAP parameters set by you to specify the region you wish to view.
Arguments

RLAT  Latitude of a point along the line.
RLON  Longitude of a point along the line.
IFST  A flag that is set to zero for the first point of a line and non-zero for all other points of the line.
IAM   An integer array containing an area map.
IGP   Group identifier for the line.
IDL   Area identifier for the area to the left of the line.
IDR   Area identifier for the area to the right of the line.

SUBROUTINES MAPITM and MAPIQM

Purpose  MAPGRM uses the routines MAPITM and MAPIQM to draw lines of latitude and longitude. The lines are masked by the area map created by MAPBLA. You may call MAPITM directly to draw the projection of an arbitrary line, defined by the latitudes and longitudes of points along it, masked by the area map.

Usage  CALL MAPITM (RLAT, RLON, IFST, IAM, XCS, YCS, MCS, IAI, IAG, MAI, LPR)
CALL MAPIQM (IAM, XCS, YCS, MCS, IAI, MAI, LPR)

You must call MAPITM once for each point along the line. After the last call to MAPITM for a given line, you must call MAPIQM to signal the end of the line.

The projection of the line segment joining two points on the globe is considered to be the straight-line segment joining the projections of the points; no attempt is made to project it as if it were a portion of a great circle.

- If both endpoints of a line segment are visible, the segment is considered to be entirely visible.
- If both endpoints are invisible, the segment is considered to be entirely invisible.
- If one endpoint is visible and the other is not, a new point is interpolated at the boundary between the visible and invisible portions.

Only the visible portions of the line are drawn in the area map.

Note: Because of the way MAPITM and MAPIQM work, (as described above) points defining a line should not be too far apart on the globe.

Boundaries: There are two types of boundaries between visible and invisible regions:

- The limb is a boundary between a projectable region and an unprojectable one. The limb may be circular, elliptical, or some other shape, depending on the projection being used. For example, an
orthographic projection has as its limb a circle, centered at (0,0), with a radius of 1.

- The *perimeter* is a rectangular or elliptical boundary defined by EZMAP parameters set by you to specify the region you wish to view.

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLAT</td>
<td>Latitude of a point along the line.</td>
</tr>
<tr>
<td>Rlon</td>
<td>Longitude of a point along the line.</td>
</tr>
<tr>
<td>IFST</td>
<td>A flag that is set to zero for the first point of a line and non-zero for all other points of the line.</td>
</tr>
<tr>
<td>IAM</td>
<td>Integer array in which an area map has been constructed by MAPBLA.</td>
</tr>
<tr>
<td>XCS, YCS</td>
<td>Real arrays, of dimension MCS, to be used by MAPGRM in calls to the user-provided routine LPR.</td>
</tr>
<tr>
<td>MCS</td>
<td>Dimension of the arrays XCS and YCS. MCS must be greater than or equal to 2; the value 100 is suggested.</td>
</tr>
<tr>
<td>IAI, IAG</td>
<td>Integer arrays, of dimension MAI, to be used by MAPGRM in calls to the user-provided routine LPR.</td>
</tr>
<tr>
<td>MAI</td>
<td>Dimension of the arrays IAI and IAG. MAI must be greater than or equal to 2. If other groups of boundary lines are added to the area map (by calling MAPITA and MAPIQA or by calling AREDAM), then MAI must be increased by the number of such groups.</td>
</tr>
<tr>
<td>LPR</td>
<td>A line-processing routine, which is called once for each piece of the masked grid. Each such piece is contained in exactly one area defined by the area map. LPR must appear in an EXTERNAL statement in the routine that calls MAPGRM. The routine LPR must be provided by you and it must have the following structure:</td>
</tr>
</tbody>
</table>

```fortran
SUBROUTINE LPR (XCS,YCS,NCS,IAI,IAG,NAI)
DIMENSION XCS(*),YCS(*),IAI(*),IAG(*)

(CODE TO PROCESS POLYLNE DEFINED BY XCS, YCS, IAI, AND IAG)

RETURN
END
```

XCS, YCS Hold the X and Y coordinates, in the fractional coordinate system, of NCS points defining a piece of the grid.

NCS Number of X and Y coordinates in the arrays XCS and YCS.

IAI, IAG Hold NAI pairs of identifiers for the area within which the piece of the polyline lies. For each value of I from 1 to NAI, IAI(I) is the area identifier for the area with respect...
to the group of edges specified by the group identifier IAG(I).

NAI Number of values in IAI and IAG. NAI will be equal to the number of groups of edges that you put in the area map.

You can assume that MAPGRM has done the following call:

CALL SET (0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,1.0)

This call ensures that, if the normalized device coordinates in XCS and YCS are used in calls to such routines as GPL, the results will be correct. LPR may do its own SET call to achieve some other effect.

Example This example uses the routines in the package EZMAPA to produce a colored map of a Mercator projection of the globe, showing world political boundaries, with lines of latitude and longitude drawn only over water. It should be noted that this example will work properly only on a device with a color overlay capability. If you attempt to send the output to a device like the color DICOMED, the attempt to draw black lines will not work.

PROGRAM COLRIT
C Define the array that holds the area map.
C
DIMENSION IAM(250000)
C
C Dimension the arrays needed by ARSCAM for edges.
C
DIMENSION XCS(10000),YCS(10000)
C
C Dimension the arrays needed by ARSCAM and ARDRLN for area and group ids.
C
DIMENSION IAI(10),IAG(10)
C
C Define the RGB triples needed below.
C
DIMENSION RGB(3,14)
C
DATA RGB / 0.70,0.70,0.70 , 0.75,0.50,1.00 , 0.50,0.00,1.00 , + 0.00,0.00,1.00 , 0.00,0.50,1.00 , 0.00,1.00,1.00 , + 0.00,1.00,0.60 , 0.00,1.00,0.00 , 0.70,1.00,0.00 , + 1.00,1.00,0.00 , 1.00,0.75,0.00 , 1.00,0.38,0.38 , + 1.00,0.00,0.38 , 1.00,0.00,0.00 /
C
C Define a set of indices determining the ordering of the colors.
C
DIMENSION IOC(14)
C
DATA IOC / 6,2,5,12,10,11,1,3,4,8,9,7,13,14 /
C Define an array for aspect source flags.
C
DIMENSION IF(13)
C
C Declare the routine which will color the areas and the one which
C will draw the lines of latitude and longitude over water.
C
EXTERNAL COLRA,M,COI

C Open GKS.
C
CALL OPN

C Re-set certain aspect source flags to "individual".
C
CALL QASF (IE,IF)
IF(11)=1
IF(12)=1
CALL GSASF (IF)

C Force solid fill.
C
CALL GSFAIS (1)

C Define 15 different color indices. The first 14 are spaced through
C the color spectrum and the final one is black.
C
CALL SETUSV ('IM',15)

DO 101 J=1,14
I=IOC(J)
IF (RGB(1,I).GE.RGB(2,I).AND.RGB(1,I).GE.RGB(3,I)) THEN
IND=1
IND=2
ELSE IF (RGB(3,I).GE.RGB(1,I).AND.RGB(3,I).GE.RGB(2,I)) THEN
IND=3
END IF
CALL SETUSV ('IR',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
CALL SETUSV ('IG',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
CALL SETUSV ('IB',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
101 CONTINUE

CALL SETUSV ('IR',1)
CALL SETUSV ('IG',1)
CALL SETUSV ('IB',1)
CALL SETUSV ('IN',0)

C Set up EZMAP, but don't draw anything.
C
CALL MAPSTC ('OU','PO')
CALL MAPROJ ('ME',O.,O.,O.)
CALL MAPSET ('MA',O.,O.,O.,O.)
Set the number of vertical strips and the group identifiers to be used by MAPBLA.

```
CALL MAPSTI ('VS',150)
CALL MAPSTI ('G1',1)
CALL MAPSTI ('G2',2)
```

Initialize EZMAP.

```
CALL MAPINT
```

Initialize the area map.

```
CALL ARINAM (IAM,250000)
```

Add edges to the area map.

```
CALL MAPBLA (IAM)
```

Pre-process the area map.

```
CALL ARPRAM (IAM,0,0,0)
```

Compute and print the amount of space used in the area map.

```
ISU=250000-(IAM(6)-IAM(5)-1)
PRINT * , 'SPACE USED IN AREA MAP IS ',ISU
```

Set the background color.

```
CALL GSCR (1,0,1.,1.,1.)
```

Color the map.

```
CALL ARSCAM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRAM)
```

In black, draw a perimeter and outline all the countries. We turn off the labels (since they seem to detract from the appearance of the plot) and we reduce the minimum vector length so as to include all of the points in the boundaries.

```
CALL SETUSV ('II',15)
CALL MAPSTI ('LA',0)
CALL MAPSTI ('MV',1)
CALL MAPLBL
CALL MAPLOT
```

Draw lines of latitude and longitude over water. They will be in black because of the SETUSV call above.

```
CALL MAPGRM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRLN)
```

Advance the frame.
CALL FRAME
C
C Close GKS.
C
CALL CLSGKS
C
C Done.
C
STOP
C
END

SUBROUTINE COLRAM (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
C
C For each area, get a set of points (using normalized device
cordinates), two group identifiers and their associated area
cifiers. If both of the area identifiers are zero or negative,
c the area need not be color-filled; otherwise, it is filled with
c a color obtained from MAPACI. If the area is defined by more than
C 150 points, print a comment. (Assume that the device being used won't
C handle polygons defined by more points than that.)
C
IF (IAI(1).GE.0.AND.IAI(2).GE.0) THEN
  ITM=MAXO (IAI(1), IAI(2))
  IF (ITM.GT.0) THEN
    IF (NCS.GT.150) PRINT *, 'COLRAM - NCS TOO BIG - ', NCS
    CALL SETUSV (' II', MAPACI(ITM))
    CALL GFA (NCS-1, XCS, YCS)
  END IF
END IF
C
C Done.
C
RETURN
C
END

SUBROUTINE COLRLN (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
C
C For each line segment, get a set of points (using normalized
cdevice coordinates), two group identifiers and their associated
carea identifiers. If both of the area identifiers are zero or
C negative, the segment is not drawn; otherwise, use MAPACI to
c see if the segment is over water and, if so, draw the segment.
C If the segment is defined by more than 150 points, print a
C comment.
C
IF (IAI(1).GE.0.AND.IAI(2).GE.0) THEN
  ITM=MAXO (IAI(1), IAI(2))
  IF (MAPACI(ITM).EQ.1) THEN
    IF (NCS.GT.150) PRINT *, 'COLRLN - NCS TOO BIG - ', NCS
    CALL GPL (NCS, XCS, YCS)
  END IF
END IF
Important Note  These routines require a great deal of memory and CPU time. In this example, the area map was dimensioned 250,000; a little over 225,000 words were actually required. The job ran about 22 seconds on the CRAY X-MP computer. Reducing the number of vertical strips used would cut both of these numbers significantly.
GRIDAL
GRIDAL

SUBROUTINE GRIDAL (MAJRX,MINRX,MAJRY,MINRY,IXLAB,IYLAB,IGPH,X,Y)

Latest Revision August 1987

Purpose

This package allows one to draw backgrounds for X/Y plots. Included are routines for drawing grids, perimeters, and pairs of axes.

Usage

Each user entry point in this package (GACOLR, GRID, GRIDAL, GRIDL, HALFA, LABMOD, PERIM, PERIML, TICKS, and TICK4) is described below. Here, we discuss features of the package as a whole.

Each of the routines GRID, GRIDL, HALFA, PERIM, PERIML, and GRIDAL draws a background of some sort within the current GKS viewport, as follows:

GRID draws an unlabeled grid
GRIDL draws a labeled grid
HALFA draws a pair of intersecting axes
PERIM draws an unlabeled perimeter
PERIML draws a labeled perimeter
GRIDAL draws any of the above

Ticks, grid lines, and numeric labels are positioned as determined by the current GKS window and by the values of the two flags 'LS' and 'MI', in the SPPS package. (The SPPS routine SET may be called to set the GKS viewport and window and the flags 'LS' and 'MI'; the flag 'LS' determines whether the mappings of user "world" coordinates into the viewport are linear or logarithmic and the flag 'MI' whether the mappings are in the normal direction or are mirror-imaged.)

The routines GACOLR, LABMOD, TICKS, and TICK4 do no drawing. Each is called to preset parameters which affect the behavior of a subsequent call to one of the background-drawing routines, as follows:

GACOLR sets the color of background parts
LABMOD changes the appearance of labels
TICKS and TICK4 change the length and direction of ticks

Arguments

Several of the routines have arguments MJRX, MNRX, MJRY, and MNRY, which specify the number of major and minor divisions of the horizontal (X) and vertical (Y) axes of the current viewport. These parameters have different meanings, depending on the current setting of the linear/log flag 'LS' in SPPS.

If the axis is linear: MJRX(Y) specifies the number of major divisions of the X(Y) axis and MNRX(Y) specifies the number of minor divisions within each major division. In each case, the value specifies the number of spaces between grid lines or ticks rather than the number of lines or ticks. There is always one more major division line or mark than the
number of major divisions specified by MJRX(Y). Similarly, there is always one less minor division line or tick per major division than the number of minor divisions per major division specified by MNRX(Y).

If the axis is logarithmic: Each major division point occurs at a value $10^{**MJRX(Y)}$ times the previous point. For example, if the minimum X-axis value were 3., the maximum X-axis value 3000. and MJRX 1, then the major division points would be 3., 30., 300., and 3000. If MNRX(Y).LE.10, there are nine minor divisions within each major division. For example, between 3. and 30., there would be minor division points at 6., 9., 12., . . . 27. If MNRX(Y).GT.10, minor divisions are omitted.

**Entry Points**
- GALBEX, GRID, GRIDAL, GRIDL, HALFAX, LABMOD, PERIM, PERIML, TICKS, TICK4

**Common Blocks**
- GAREIN and GACHAR.

**Required Routines**
- SETER and the SPPS.

**Required GKS Level**
- 0A

**I/O**
- Plots backgrounds.

**Precision**
- Single.

**Language**
- FORTRAN 77.

**History**
- These routines have been a part of the NCAR System Plot Package for many years. As part of the GKS effort, they were incorporated into a separate package. The code here represents a total rewrite in October 1986.

**SUBROUTINE GACOLR**

**Purpose**
- To set the color of various parts of the background.

**Usage**
- CALL GACOLR (KAXS,KLBL,KMJT,KMNT)

**Arguments**
- KAXS, KLBL, KMJT, and KMNT are the color indices of the desired colors of the axes, the labels, the major ticks/grid lines, and the minor ticks/grid lines, respectively. Values less than or equal to zero imply that no call is to be done to set the color before drawing items of the associated type. The default value of each of the four parameters is zero.
SUBROUTINE GRID

Purpose To draw an unlabeled grid.

Usage CALL GRID (MJRX, MNRX, MJRY, MNRY)

Arguments See the package description, above.

SUBROUTINE GRIDAL

Purpose A general entry point for all supported types of backgrounds. Each of the other background-drawing routines is implemented by a call to GRIDAL.

Usage CALL GRIDAL (MJRX, MNRX, MJRY, MNRY, IXLB, IYLB, IGPH, XINT, YINT)

Arguments MJRX, MNRX, MJRY, and MNRY specify the major and minor divisions of the two axes, as described in the package description, above.

IXLB and IYLB are integer flags, defined as follows:

\[
\begin{align*}
\text{IXLB} = -1 & \quad \text{No X axis drawn.} \\
& \quad \text{No X-axis labels.} \\
\text{IXLB} = 0 & \quad \text{X axis drawn.} \\
& \quad \text{No X-axis labels.} \\
\text{IXLB} = 1 & \quad \text{X axis drawn.} \\
& \quad \text{X-axis labels.} \\
\text{IYLB} = -1 & \quad \text{No Y axis drawn.} \\
& \quad \text{No Y-axis labels.} \\
\text{IYLB} = 0 & \quad \text{Y axis drawn.} \\
& \quad \text{No Y-axis labels.} \\
\text{IYLB} = 1 & \quad \text{Y axis drawn.} \\
& \quad \text{Y-axis labels.}
\end{align*}
\]

IGPH specifies the background type:

<table>
<thead>
<tr>
<th>IGPH</th>
<th>X axis</th>
<th>Y axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GRID</td>
<td>GRID</td>
</tr>
<tr>
<td>1</td>
<td>GRID</td>
<td>PERIM</td>
</tr>
<tr>
<td>2</td>
<td>GRID</td>
<td>HALFAX</td>
</tr>
<tr>
<td>4</td>
<td>PERIM</td>
<td>GRID</td>
</tr>
<tr>
<td>5</td>
<td>PERIM</td>
<td>PERIM</td>
</tr>
<tr>
<td>6</td>
<td>PERIM</td>
<td>HALFAX</td>
</tr>
<tr>
<td>8</td>
<td>HALFAX</td>
<td>GRID</td>
</tr>
<tr>
<td>9</td>
<td>HALFAX</td>
<td>PERIM</td>
</tr>
<tr>
<td>10</td>
<td>HALFAX</td>
<td>HALFAX</td>
</tr>
</tbody>
</table>

XINT and YINT are the user "world" coordinates of the point of intersection of the two axes if IGPH equals 10. For other values of IGPH for which one of the axes is of type HALFAX, XINT and/or YINT specify the position of that axis.

August 1987 307
**SUBROUTINE GRIDL**

**Purpose**
To draw a labeled grid. Each major division is labeled with its numeric value.

**Usage**
CALL GRIDL (MJRX,MNRX,MJRY,MNRY)

**Arguments**
See the package description, above.

**SUBROUTINE HALFAX**

**Purpose**
To draw orthogonal axes intersecting at a specified point and with a specified set of labels.

**Usage**
CALL HALFAX (MJRX,MNRX,MJRY,MNRY,XINT,YINT, + IXLB,IYLB)

**Arguments**
All arguments are as defined for GRIDAL, above. In fact, the above call is equivalent to

CALL GRIDAL (MJRX,MNRX,MJRY,MNRY,IXLB,IYLB, + 10,XINT,YINT)

**SUBROUTINE LABMOD**

**Purpose**
To preset parameters controlling the appearance of labels drawn by GRIDAL, GRIDL, ... et al. LABMOD itself does no plotting and, in order to have any effect, must be called prior to the background-drawing routines for which it is presetting parameters.

**Usage**
CALL LABMOD (FMTX,FMTY,NUMX,NUMY,ISZX,ISZY, + IXDC,IYDC,IXHR)

**Arguments**
FMTX and FMTY are of type CHARACTER and contain format specifications for the X-axis and Y-axis numerical labels produced by GRIDAL, GRIDL, HALFAX, or PERIML. The specification must begin with a left parenthesis and end with a right parenthesis and must not be more than ten characters long. Conversions of types E, F, G, and I are allowed; for example, one might use FMTX='(F8.2)' and FMTY='(E10.0)'. The default for both formats is '(E10.3)'.

Note: I formats are allowed by this version of GRIDAL; they were not allowed by previous versions.

NUMX, if non-zero, is the number of characters in each X-axis numeric label; if LBLX is a string produced by the format FMTX, then the label will be the substring LBLX(1:NUMX). If NUMX is 0, then the label will
be the substring LBLX(m:n), where LBLX(m:m) is the first non-blank character in LBLX, and LBLX(n:n) is the last non-blank character following LBLX(m:m). Using a non-zero NUMX causes the labels to be centered differently than if a zero value is used. NUMY is defined similarly and applies to Y-axis labels. The default value for both parameters is 0.

ISZX and ISZY are character sizes for the labels, specified in plotter address units, just as for the SPPS routines PWRIT and WTSTR. The default value for both is 10.

IXDC is the decrement, in plotter address units, from the left edge of the current viewport to the nearest X address of the label specified by FMTY, NUMY, and ISZY. For example, if the horizontal extent of the current viewport is defined by the normalized device coordinates .1 and .9, and if IXDC is 60, and if there has been no call to the SPPS routine SETI, then labels on the Y axis will end at plotter coordinate 43 (.1*1023+1-60). Negative values may be used to put labels on the other side of the viewport; in the example given, changing IXDC to -878 (-.8*1023-60) would put the labels on the right side of the viewport, with their left edges 60 plotter-coordinate units away from the edge of the viewport. There are two special values of IXDC:

If IXDC=0, the Y-axis labels will end 20 plotter address units to the left of the viewport (equivalent to using IXDC=20).

If IXDC=1, Y-axis labels will begin 20 plotter address units to the right of the viewport (equivalent to using IXDC=-20-w, where w is the width of the viewport, in plotter address units).

The default value of the X decrement is 20.

When HALFAX is called or when GRIDAL is called with IGPH = 2, 6, or 10, IXDC is the distance from the Y axis, rather than from the minimum viewport coordinate, and the special values 0 and 1 are equivalent to 20 and -20.

IYDC is the decrement, in plotter address units, from the bottom edge of the current viewport to the nearest Y address of the label specified by FMTX, NUMX, and ISZX. Note that negative values may be used to put labels above the viewport. There are two special values of IYDC:

If IYDC=0, the top of the X-axis labels will be 20 plotter address units below the bottom edge of the viewport (equivalent to using IYDC=20).

If IYDC=1, the bottom of the X-axis labels will be 20 plotter address units above the top edge of the viewport (equivalent to using IYDC=20-h, where h is the height of the viewport, in plotter address units).

The default value of the Y decrement is 20.
When HALFAX is called or when GRIDAL is called with IGPH = 8, 9, or 10, IYDC is the distance from the X axis, rather than from the minimum viewport coordinate, and the special values 0 and 1 are equivalent to 20 and -20.

IXOR specifies the orientation of the X-axis labels:

IXOR = 0  horizontal
= 1  vertical

The default orientation is horizontal.

**SUBROUTINE PERIM**

**Purpose**  To draw an unlabeled perimeter with inward-pointing tick marks. The directions and lengths of tick marks may be changed by calling TICKS and/or TICK4 (see below).

**Usage**  CALL PERIM (MJRX,MNRX,MJRY,MNRY)

**Arguments**  See the package description, above.

**SUBROUTINE PERIML**

**Purpose**  To draw a labeled perimeter with inward-pointing tick marks. The directions and lengths of tick marks may be changed by calling TICKS and/or TICK4 (see below).

**Usage**  CALL PERIML (MJRX,MNRX,MJRY,MNRY)

**Arguments**  See the package description, above.

**SUBROUTINE TICKS**

**Purpose**  To allow program control of tick mark length and direction. This routine has been superseded by TICK4, which should be used instead.

**Usage**  CALL TICKS (LMJR,LMNR)

**Arguments**  The above call is equivalent to

CALL TICK4 (LMJR,LMNR,LMJR,LMNR)

See the description of TICK4, below.
SUBROUTINE TICK4

Purpose  To allow program control of tick mark length and direction.

Usage  CALL TICK4 (LMJX,LMNX,LMJY,LMNY)

Arguments  LMJX and LMNX are the lengths, in plotter address units, of major and minor ticks on the X axis. The default values are 12 and 8.

LMJY and LMNY are the lengths, in plotter address units, of major and minor ticks on the Y axis. The default values are 12 and 8.

By default, tick marks point inward. Negative values of LMJX, LMNX, LMJY, and LMNY may be used to create outward-pointing tick marks.
HAFTON

SUBROUTINE HAFTON (Z,L,M,N,FLO,HI,NLEV,NOPT,NPRM,ISPV,SPVAL)

Dimension of Arguments
Z(L,M)

Latest Revision August 1987

Purpose HAFTON draws a halftone picture from data stored in a rectangular array with the intensity in the picture proportional to the data value.

Usage If the following assumptions are met, use
CALL EZHFTN (Z,M,N)

Assumptions:
- All of the array is to be drawn.
- Lowest value in Z will be at lowest intensity on reader/printer output.
- Highest value in Z will be at highest intensity.
- Values in between will appear linearly spaced.
- Maximum possible number of intensities are used.
- The picture will have a perimeter drawn.
- FRAME will be called after the picture is drawn.
- Z is filled with numbers that should be used (no missing values).

If these assumptions are not met, use
CALL HAFTON (Z,L,M,N,FLO,HI,NLEV,NOPT,NPRM,ISPV,SPVAL)

ARGUMENTS

On Input for EZHFTN
Z M by N array to be used to generate a halftone plot.
M First dimension of Z.
N Second dimension of Z.

On Output for EZHFTN
All arguments are unchanged.

On Input for HAFTON
Z The origin of the array to be plotted.
L The first dimension of Z in the calling program.
M The number of data values to be plotted in the x-direction (the first subscript direction). When plotting all of an array, L = M.
N The number of data values to be plotted in the y-direction (the second subscript direction).
**FLO** The value of Z that corresponds to the lowest intensity. (When NOPT.LT.0, FLO corresponds to the highest intensity.) If FLO=HI=0.0, MIN(Z) will be used for FLO.

**HI** The value of Z that corresponds to the highest intensity. (When NOPT.LT.0, HI corresponds to the lowest intensity.) If HI=FLO=0.0, MAX(Z) will be used for HI.

**NLEV** The number of intensity levels desired. 16 maximum. If NLEV = 0 or 1, 16 levels are used.

**NOPT** Flag to control the mapping of Z onto the intensities. The sign of NOPT controls the directness or inverseness of the mapping.

- **NOPT** positive yields direct mapping. The largest value of Z produces the most dense dots. On mechanical plotters, large values of Z will produce a dark area on the paper. With the film development methods used at NCAR, large values of Z will produce many (white) dots on the film, also resulting in a dark area on reader-printer paper.

- **NOPT** negative yields inverse mapping. The smallest values of Z produce the most dense dots resulting in dark areas on the paper.

The absolute value of NOPT determines the mapping of Z onto the intensities. For IABS(NOPT)

- **= 0** The mapping is linear. For each intensity there is an equal range in Z value.

- **= 1** The mapping is linear. For each intensity there is an equal range in Z value.

- **= 2** The mapping is exponential. For larger values of Z, there is a larger difference in intensity for relatively close values of Z. Details in the larger values of Z are displayed at the expense of the smaller values of Z.

- **= 3** The mapping is logarithmic, so details of smaller values of Z are shown at the expense of larger values of Z.

- **= 4** Sinusoidal mapping, so mid-range values of Z show details at the expense of extreme values of Z.

- **= 5** Arcsine mapping, so extreme values of Z are shown at the expense of mid-range values of Z.

**NPRM** Flag to control the drawing of a perimeter around the halftone picture.

- **NPRM=0**: The perimeter is drawn with ticks pointing at data locations. (Side lengths are proportional to number of data values.)

- **NPRM positive**: No perimeter is drawn. The picture fills the frame.

- **NPRM negative**: The picture is within the confines of the user's current viewport setting.
ISPV Flag to tell if the special value feature is being used. The special value feature is used to mark areas where the data is not known or holes are wanted in the picture.

- ISPV = 0: Special value feature not in use. SPVAL is ignored.
- ISPV non-zero: Special value feature in use. SPVAL defines the special value. Where Z contains the special value, no halftone is drawn. If ISPV
  = 0 Special value feature not in use. SPVAL is ignored.
  = 1 Nothing is drawn in special value area.
  = 2 Contiguous special value areas are surrounded by a polygonal line.
  = 3 Special value areas are filled with X(s).
  = 4 Special value areas are filled in with the highest intensity.

SPVAL The value used in Z to denote missing values. This argument is ignored if ISPV = 0.

On Output for HAFTON All arguments are unchanged.

Note This routine produces a huge number of plotter instructions per picture, averaging over 100,000 line-draws per frame when M = N.

Entry Points EZHFTN, HAFTON, ZLSET, GRAY, BOUND, HFINIT

Common Blocks HAFT01, HAFT02, HAFT03, HAFT04

Required Library Routines

Required GKS Level 0A

I/O Plots halftone picture.

Precision Single

Language FORTRAN 77

History Rewrite of PHOMAP originally written by M. Perry of High Altitude Observatory, NCAR.

August 1987
**Algorithm** Bi-linear interpolation on plotter (resolution-limited) grid of normalized representation of data.

**Portability** ANSI FORTRAN 77.

<table>
<thead>
<tr>
<th>Internal Parameters</th>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XLT</td>
<td>0.1</td>
<td>Left-hand edge of the plot when NSET=0. (0.0=left edge of frame, 1.0=right edge of frame.)</td>
</tr>
<tr>
<td></td>
<td>YBT</td>
<td>0.1</td>
<td>Bottom edge of the plot when NSET=0. (0.0=bottom of frame, 1.0=top of frame.)</td>
</tr>
<tr>
<td></td>
<td>SIDE</td>
<td>0.8</td>
<td>Length of longer edge of plot (see also EXT).</td>
</tr>
<tr>
<td></td>
<td>EXT</td>
<td>.25</td>
<td>Lengths of the sides of the plot are proportional to M and N (when NSET=0) except in extreme cases, namely, when MIN(M,N)/MAX(M,N) is less than EXT. Then a square plot is produced. When a rectangular plot is produced, the plot is centered on the frame (as long as SIDE+2<em>XLT = SIDE+2</em>YBT=1., as with the defaults.)</td>
</tr>
<tr>
<td></td>
<td>ALPHA</td>
<td>1.6</td>
<td>A parameter to control the extremeness of the mapping function specified by NOPT. (For IABS(NOPT)=0 or 1, the mapping function is linear and independent of ALPHA.) For the nonlinear mapping functions, when ALPHA is changed to a number closer to 1., the mapping function becomes more linear; when ALPHA is changed to a larger number, the mapping function becomes more extreme.</td>
</tr>
<tr>
<td></td>
<td>MXLEV</td>
<td>16</td>
<td>Maximum number of levels. Limited by plotter.</td>
</tr>
<tr>
<td></td>
<td>NCRTG</td>
<td>8</td>
<td>Number of CRT units per gray-scale cell. Limited by plotter.</td>
</tr>
<tr>
<td></td>
<td>NCRTF</td>
<td>1024</td>
<td>Number of plotter address units per frame.</td>
</tr>
<tr>
<td></td>
<td>IL</td>
<td>(Below)</td>
<td>An array defining which of the available intensities are used when less than the maximum number of intensities are requested.</td>
</tr>
<tr>
<td>NLEV</td>
<td>Intensities Used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5, 11,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4, 8, 12,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3, 6, 10, 13,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2, 5, 8, 11, 14,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1, 4, 7, 9, 12, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1, 4, 6, 8, 10, 12, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1, 3, 5, 7, 9, 11, 13, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1, 3, 4, 6, 8, 10, 12, 13,15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1, 3, 4, 6, 7, 9, 10, 12, 13, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1, 2, 3, 5, 6, 8, 10, 11, 13, 14, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HISTGR
HISTGR

SUBROUTINE HISTGR (DAT1,NDIM,NPTS,IFLAG,CLASS,NCLASS,WRK,NWRK)

Latest Revision  August 1987

Purpose  HISTGR plots a histogram with various options including specification of
class values, spacing between histogram bars, shading of bars, windowing
(i.e. scaling), specification of color, labels, titles, etc. Data values are par-
titioned into classes; histogram bars represent either number of oc-
currences within each class, or a Y-value associated with that class (user
choice). Options are set by calls to subroutines HSTOPL, HSTOPR,
HSTOPC, and HSTOPI before the call to HISTGR.

Usage  To draw a basic histogram (one utilizing the default features):

CALL HISTGR (DAT1,NDIM,NPTS,IFLAG,CLASS,
+   NCLASS,WRK,NWRK)

Error handling is performed by the ERPRT77 centralized error-handling
package. To recover from a warning (recoverable error), call
ENTSR(IDUM,1) before calling HISTGR. Otherwise an error message is
printed and the run is terminated.

ARGUMENTS

On Input  DAT1  Two dimensional real array containing data of one of two
types, either values to be collected into class intervals before
plotting, or values which have already been assigned to class
intervals and only need to be displayed. See argument
IFLAG for a more complete description of HISTGR input
data options. DAT1 is dimensioned: DAT1(NDIM,2).

NDIM  The size of the first dimension of DAT1 as set in the dimen-
statement of the calling program.

NPTS  Number of values actually stored into DAT1 on this call.
NPTS must always be less than or equal to NDIM.

IFLAG  An integer flag which selects one of four options provided by
the HISTGR utility. The options are:

IFLAG = 0  A single array of length NPTS is loaded into
the DAT1 array. HISTGR computes NCLASS
equally sized class intervals that vary from the
minimum value in DAT1 to the maximum
value in steps of (MAX-MIN)/NCLASS.

All values of DAT1 that fall in each class inter-
val are separately accumulated for that inter-
val. The final tabulations are plotted as a his-
togram of NCLASS bars. The bar height can
be labeled with the number of points that fall
within this particular class interval (bin size),
or it can be given as a percentage of the number of values input, NPTS.

Note that under this option the user has no control over the range of the class intervals. They are internally determined from the range of the data.

**IFLAG = 1** This option is similar to the IFLAG = 0 option except that the user can select the range of the class intervals into which the data are collected. For example, say the user wants to collect the number of occurrences of the DAT1 values that fall within 5 equally spaced intervals in the value range from 0. to 10. The user would then input NCLASS+1 class interval end points into array CLASS, namely 0., 2., 4., 6., 8., and 10. These values need not be entered in monotonically increasing order and need not be equally spaced.

**IFLAG = 2** This option allows the user to enter and display data which has already been accumulated into class intervals, i.e., already available histograms. The data input to DAT1 thus have percentage of total, or number of occurrences values. In this case the number of points in DAT1, NPTS, is equal to the number of class intervals (histogram bars), NCLASS. The NCLASS class interval midpoints are loaded into array CLASS. They do not have to be of equal width.

**IFLAG = 3** This option is the same as option IFLAG = 2 except that two histograms can be displayed for comparison purposes. The first histogram is loaded into DAT1(NPTS,1). The second histogram is loaded into DAT1(NPTS,2). The first histogram can partially shade or obscure the second histogram by the appropriate selection of the SPAC and OVERLP options.

Note that NPTS = NCLASS when IFLAG = 2 or 3.

**CLASS** Real array containing class values, dimensioned (NCLASS+1). This array has the following IFLAG dependencies:

**IFLAG = 0** CLASS is not used.

**IFLAG = 1** NCLASS+1 class interval end points are loaded into array CLASS in a monotonically increasing order. The intervals need not be of equal width.
**HISTGR**

**IFLAG = 2** NCLASS midpoint intervals are loaded into array CLASS. They must be in monotonically increasing order, but need not be of equal widths. The histogram bars will however be displayed with equal widths.

**IFLAG = 3** Same as for IFLAG = 2.

- **NCLASS** Number of class intervals (histogram bars) specified. NCLASS must be .GE. 1.
- **WRK** Real scratch array, dimensioned by NWRK in the dimension statement of the calling program.
- **NWRK** The dimension size of array WRK determined from: NDIM + 3 * (NCLASS + 1)

Note: The frequency axis label values will be integers if the maximum Y-value (calculated within HISTGR) is an integer multiple of 4. Otherwise, real values with format F11.1 are used.

**On Output** All arguments remain unchanged except the scratch array WRK.

**Entry Points** HSTBKD, HSTEXP, HSTMED, HSTOPC, HSTOPL, HSTOPL, HSTOPI, HSTSTR, HISTGR, HSTLST, NWTSTR

**Common Blocks** HSTGC1, HSTGC2

**I/O** Plots histograms using various options.

**Precision** Single

**Required Library Files** The ERPRT77 package and the SPPS.

**Required GKS Level** 0A

**Language** FORTRAN 77

**History** First written in the fall of 1984. Revised summer of 1987.

**Algorithm** Calculates occurrence of Y-values within classes, or assigns Y-values to X-values (class mid-values), depending on which flag is chosen by the user. If Y-values are assigned to classes, class interval boundaries are handled as follows. Y-values which fall on the left boundary of a class interval are counted in that interval. This accounts for all Y-values at an interval boundary except for the right boundary of the rightmost class interval. Such occurrences are added to the rightmost interval, thus only this interval contains Y-values which fall on either boundary. Draws a histogram incorporating various user-defined options.
SUBROUTINE HSTOPL

Purpose To set various options for the package.

Usage CALL HSTOPL (STRING)

Arguments STRING A character string (which may have up to seven characters) where the first 2 or 3 characters are abbreviations for the option followed by '=' followed by 'ON' or 'OFF'. 'OFF' may be abbreviated to 'OF'. Embedded spaces are not allowed.

Example: CALL HSTOPL('PER=OFF')

The following options are turned 'ON' or 'OFF' by this routine. The default values are listed at the end of the HISTGR documentation.

HOR Horizontal. The histogram bars are drawn horizontally if 'HOR=ON'. If 'HOR=OFF', they are drawn vertically.

PER Percent axis. If 'PER=ON', a labeled percent axis is drawn on the right side of the histogram (or on top if horizontal). If 'PER=OFF', the percent axis is not drawn.

MID Midvalues. If 'MID=ON', the class labels are put at the midpoint of each interval. If 'MID=OFF', the class interval end points are labeled. MID defaults to ON if IFLAG = 2 or 3.

SHA Shading. If 'SHA=ON', the histogram rectangles are shaded. If 'SHA=OFF', the bars are not shaded. Laser printers may vary as to how they interpret shading; some will shade the bars black, others will only draw the outline, etc., depending on whether or not they support FILL AREA in hardware. On color terminals, the shading will correspond to whatever the RGB color table assigns to color index 1, even when the color option is 'COL=OFF' (and all other portions of the histogram are white). Terminals that do not support FILL AREA or color will show the outline of the "shaded" histogram bars in white (they will appear to be unshaded).
HISTGR

DRL Draw lines.
If 'DRL=ON', lines are drawn through the histogram rectangles where Y-axis tick marks would occur. When IFLAG = 3 (when comparing two datasets in one histogram), lines are drawn through the bars associated with the first dataset only. If 'DRL=OFF', lines are not drawn.

MED Median.
If 'MED=ON', a line is drawn through the median of all points. If 'MED=OFF', this line is not drawn. MED does not apply when assigning Y-values to X-values; it is valid only for IFLAG = 1 or 2.

PRM Perimeter.
If 'PRM=ON', a perimeter is drawn around the histogram. If 'PRM=OFF', no perimeter is drawn.

FRA Frame advance.
If 'FRA=ON', the frame is advanced automatically after the histogram is drawn. If 'FRA=OFF', the frame is not advanced, and the user must call FRAME.

LIS List options.
If 'LIS=ON', all the options along with their values are printed on the standard output. If 'LIS=OFF', nothing is printed on the standard output.

DEF Global default.
If 'DEF=ON', all the options are set to their default values; see list of default values below. 'DEF=OFF' has no effect.

SUBROUTINE HSTOPR

Purpose To specify various REAL arrays to be used by the HISTGR package.

Usage CALL HSTOPR (STRING,ARRAY,LARR)

Arguments STRING A character string specifying the option to be set. Valid options are as follows:

'WIN=ON' or 'WIN=OFF'
'SPA=ON' or 'SPA=OFF'

WIN defines the portion of the frame that will receive the histogram. SPA determines the spacing between histogram bars.

ARRAY A real array of length LARR.

LARR Dimension of ARRAY.
The following arrays may be defined by this routine:

Windowing:

```
STRING is 'WIN=ON'
LARR = 4
ARRAY(1) = XMIN
ARRAY(2) = XMAX
ARRAY(3) = YMIN
ARRAY(4) = YMAX
```

Assumptions: These coordinates define a rectangular region of the total frame where the current histogram is to be plotted. The range of allowed values is as follows:

\[ \begin{align*}
0. &< XMIN < XMAX < 1. \\
0. &< YMIN < YMAX < 1. 
\end{align*} \]

For example, XMIN=0., XMAX=.5, YMIN=.5, and YMAX=1. would specify the upper left quadrant of the frame.

If 'WIN=OFF', the default window for the histogram will be the entire frame.

Example:

```
REAL WINDOW(4)
DATA WINDOW / .3, .7, .3, .7 /
CALL HSTOPR ('WIN=ON', WINDOW, 4)
```

Spacing:

```
STRING is 'SPA=ON'
LARR = 2
ARRAY(1) = SPAC
ARRAY(2) = OVERLP
```

SPAC — Real value used to set spacing of histogram bars; valid values are 0.0 (no spacing) to 4.0 (maximum spacing). Default spacing is SPAC = 2.0. If 'SPA=OFF', the result is the same as if SPAC = 0.0, and the value supplied in ARRAY(1) is ignored by HISTGR. Lines will be drawn around the histogram bars when 'SPA=OFF' by default. These may not be visible if 'SHA=ON' and the first color index is set to white in the color table; it is a good idea to set color index 1 to gray if 'SPA=OFF', and 'SHA=ON' (for terminals without color capability, this does not apply; see notes on 'SHA' above). If IFLAG = 3, there is a minimum amount of spacing supplied, even if SPAC = 0.0, to allow room for the second dataset histogram bars.

OVERLP — Real value used to set overlap of adjacent histogram bars when comparing two datasets in one histogram; valid values are -4.0
(maximum overlap) to 4.0 (little or no overlap). OVERLP applies only when IFLAG = 3. Default overlap is OVERLP = 0.0. If 'SPA=OFF', OVERLP is ignored by HISTGR. If no overlap is desired, set OVERLP to 4.0, and SPAC to 3.0 or greater.

Example:

    REAL ARRAY (2)
    IFLAG = 3
    ARRAY(1) = 2.0
    ARRAY(2) = -1.5
    CALL HSTOPR ('SPA=ON', ARRAY, 2)

The above example would cause two sets of histograms to have overlapping bars for comparison purposes.

SUBROUTINE HSTOPC

Purpose To specify various CHARACTER variables to be used by the HISTGR package.

Usage CALL HSTOPC (STRING,STRNG2,NUMBER,ILCH)

Arguments

    STRING A character string specifying which option is to be set. Valid options are as follows:
    'FOR=ON' or 'FOR=OFF'
    'TIT=ON' or 'TIT=OFF'
    'LAB=ON' or 'LAB=OFF'
    'FQN=ON' or 'FQN=OFF'
    'CHR=ON' or 'CHR=OFF'

By choosing the ON form for an option, the user can override the default setting of that option. A subsequent call to HSTOPC with the OFF form for an option returns the option to the default setting. All defaults are listed at the end of the HISTGR documentation.

    STRNG2 A character string up to 45 characters long.

This argument depends upon the ON form being selected for an option. For example, if 'TIT=ON', a main title is input through argument STRNG2.

    NUMBER Integer variable, which applies to the following three options:
    'FOR=ON' The maximum number of class intervals (histogram bars) that will be labeled.

    'FOR=OFF' Defaults to 9 labels for vertical bars and to 15 labels for horizontal bars.
'CHR=ON' Must be set to NCLASS, an argument of the subsequent call to be made to HISTGR.

NUMBER is not used under any other option setting.

Calls to HSTOCP with both 'FOR=ON' and 'CHR=ON' may be performed in any order; the parameters set by NUMBER are mutually exclusive in either case.

**ILCH**
An integer variable specifying the number of characters in each label of a class interval (histogram bar). This argument is only used with the 'CHR=ON' option. ILCH cannot be greater than 15.

The following options are defined by this subroutine:

**FOR** Format for class labels.
The 'FOR=OFF' default is '(G10.3)'. Although class values are real numbers, integer formats are allowed, in which case HISTGR will convert from real to integer before plotting labels.

**TIT** A main title.
The 'TIT=OFF' default is no title.

**LAB** A label for class interval (histogram bar) axis.
The 'LAB=OFF' default value is 'CLASS INTERVALS' when the HSTOPL option 'MID=OFF' is selected, and 'CLASS MID-VALUES' otherwise.

In order to delete this axis label, select 'LAB=ON' for STRING and 'NOLABEL' for STRNG2.

**FQN** The frequency axis label.
The 'FQN=OFF' default value is 'FREQUENCY'.

In order to delete this axis label, select 'FQN=ON' for STRING and 'NOLABEL' for STRNG2.

**CHR** Character labels.
Use a character string containing ILCH*NUMBER characters to specify alphanumeric class intervals (histogram bars). This is a packed string of ILCH characters per class interval label.

The character string must contain NUMBER (=NCLASS) class interval labels, even though all may not be used. See the definition of argument NUMBER.

**Example**
```
PARAMETER (NCLASS=12, ILCH=3)
CHARACTER*27 LABEL
CALL HSTOCP ('TIT=ON', 'MONTHLY PRECIPITATION in 1987', 12, 3)
LABEL = 'JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'
CALL HSTOCP ('CHR=ON', LABEL, 12, 3)
CALL HSTOCP ('FOR=ON', '(F3.0)', 12, 3)
```

In the above example, there will be 12 alphanumeric class labels, each containing 3 characters to specify the months of the year. The title indi-
cates the histogram will depict monthly precipitation in 1987.

The call to HSTOPC with 'FOR=ON' and NUMBER = 12 overrides the
default number (9) of labels that would be plotted. Note that the (F3.0)
format is ignored because 'CHR=ON'.

SUBROUTINE HSTOPI

Purpose To specify various INTEGER variables to be used by the HISTGR pack-
age.

Usage CALL HSTOPI (STRING, PARAM1, PARAM2, ICOL, LCOL)

Arguments STRING A character string specifying which option is to be set.
Valid options are as follows:

'COL=ON' or 'COL=OFF'
'CLA=ON' or 'CLA=OFF'

By choosing the ON form for an option, the user can over-
ride the default setting of that option. A subsequent call to
HSTOPI with the OFF form for an option returns the option
to the default setting. All defaults are listed at the end of
the HISTGR documentation.

PARAM1 Integer variable used to set character height of class labels
when 'CLA=ON'; 1 = small, 2 = medium, 3 = large; default
is 2 when 'CLA=OFF'.

PARAM2 Integer variable used to set orientation of class labels, from
0 (horizontal) to 90 (vertical) degrees when 'CLA=ON'. De-
fault is 0 degrees when 'CLA=OFF'.

Not used when 'COL=ON' or 'COL=OFF'.

ICOL Integer array containing values defining color indices 1-8, for
use with 'COL=ON'.

The eight components of the plot for which color indices can
be set are as follows:

ICOL(1) = color index used for shading rectangles.
ICOL(2) = color index used for shading second set of rec-
tangles (comparing two datasets when IFLAG = 3).
ICOL(3) = color index used for rectangle outlines.
ICOL(4) = color index used for drawing axes.
ICOL(5) = color index used for drawing median line.
ICOL(6) = color index used for text output (labels).
ICOL(7) = color index used for title.
ICOL(8) = color index used for drawing perimeter.

The default color index is 1 for all (when 'COL=OFF').
If 'COL=ON', the color indices and their associated colors are as follows. (These may be changed by specifying an RGB color table prior to your call to HSTOPI.)

- 0 = BLACK
- 1 = WHITE
- 2 = RED
- 3 = GREEN
- 4 = BLUE
- 5 = CYAN
- 6 = MAGENTA
- 7 = YELLOW
- 8 = ORANGE

ICOL is not used when 'CLA=ON' or 'CLA=OFF'.

**LCOL**

Integer variable specifying length of array ICOL. ICOL must be set to 8.

LCOL is not used when 'CLA=ON' or 'CLA=OFF'.

The default values for all the options are as follows:

**HSTOPL**

- 'HOR=OFF' histogram bars will be vertical.
- 'PER = ON' a labeled percent axis is drawn on the right side of the histogram (or on top if horizontal).
- 'MID = ON' labels are placed at the midpoint of each histogram bar.
- 'SHA = ON' the histogram rectangles are shaded.
- 'DRL = OFF' lines corresponding to plotted scale values are not drawn on the histogram.
- 'MED = OFF' a line denoting the median of all points is not drawn on the histogram.
- 'PRM = OFF' a perimeter around the histogram is not drawn.
- 'FRA = ON' a frame advance will occur after the call to HISTGR.
- 'LIS = OFF' nothing is printed on the standard output unit.

**HSTOPR**

- 'WIN = OFF' the histogram will be drawn within the maximum viewport of 0.0 to 1.0 in both the horizontal and vertical dimensions.
- 'SPA = ON' default values: spacing = 2.0, overlap = -1.0 This will yield spacing between bars on a single histogram and overlap of bars in the comparison of two histograms.

**HSTOPC**

- 'FOR = OFF' class label format = '{G10.3}'.
- 'TIT = OFF' no title.
- 'LAB = OFF' label = 'CLASS INTERVALS' or 'CLASS MID-VALUES' if 'MID=ON'.
- 'FQN = OFF' frequency label = 'FREQUENCY'.
- 'CHR = OFF' numeric (real or integer) class labels.

**HSTOPI**

- 'COL = OFF' color index 1 is used for all output.
- 'CLA = ON' default values: 2 = medium, 0 = horizontal labels.
ISOSRF

SUBROUTINE ISOSRF (T,LU,MU,LV,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)

Dimension of Arguments
T(LU,lv,MW),EYE(3),SLAB(MUVWP2,MUVWP2)

Latest Revision August 1987

Purpose ISOSRF draws an approximation of an iso-surface from a three-dimensional array with hidden lines removed.

Usage If the following assumptions are met, use
CALL EZISOS (T,LU,MU,lv,MV,MW,EYE,SLAB,TISO)

Assumptions:
- All of the T array is to be used.
- IFLAG is chosen internally.
- FRAME is called by EZISOS.

If the assumptions are not met, use
CALL ISOSRF (T,LU,MU,lv,lv,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)

ARGUMENTS

On Input
T Three dimensional array of data that defines the iso-surface.
LU First dimension of T in the calling program.
MU The number of data values of T to be processed in the U direction (the first subscript direction). When processing the entire array, LU = MU (and LV = MV).
LV Second dimension of T in the calling program.
MV The number of data values of T to be processed in the V direction (the second subscript direction).
MW The number of data values of T to be processed in the W direction (the third subscript direction).
EYE The position of the eye in three-space. T is considered to be in a box with opposite corners (1,1,1) and (MU,MV,MW). The eye is at (EYE(1),EYE(2),EYE(3)), which must be outside the box that contains T. While gaining experience with the routine, a good choice for EYE might be (5.0*LU,3.5*LV,2.0*MW).
MUVWP2 The maximum of (MU,MV,MW)+2; that is, MUVWP2 = MAX(MU,MV,MW)+2).
SLAB
A work space used for internal storage. SLAB must be at least MUVWP2*MUVWP2 words long.

TISO
The iso-value used to define the surface. The surface drawn will separate volumes of T that have values greater than or equal to TISO from volumes of T that have values less than TISO.

IFLAG
This flag serves two purposes.

1. First, the absolute value of IFLAG determines which types of lines are drawn to approximate the surface. Three types of lines are considered: lines of constant U, lines of constant V, and lines of constant W. The following table lists the types of lines drawn.

<table>
<thead>
<tr>
<th>IABS(IFLAG)</th>
<th>U</th>
<th>V</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>4</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>5</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>6</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>0, 7 OR MORE</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

2. Second, the sign of IFLAG determines what is inside and what is outside, hence, which lines are visible and what is done at the boundary of T. For IFLAG:

POSITIVE T values greater than TISO are assumed to be inside the solid formed by the drawn surface.

NEGATIVE T values less than TISO are assumed to be inside the solid formed by the drawn surface. If the algorithm draws a cube, reverse the sign of IFLAG.

On Output T,LU,MU,LV,MV,MW,EYE,MUVWP2,TISO and IFLAG are unchanged. SLAB has been written in.

Note
- This routine is for lower resolution arrays than ISOSRFHR. 40 by 40 by 40 is a practical maximum.
- Transformations can be achieved by adjusting scaling statement functions in ISOSRF, SET3D, and TR32.
- The hidden-line algorithm is not exact, so visibility errors can occur.
- Three-dimensional perspective character labeling of ISOSRF is possible by using the utility PWRZI. For a description of the usage, see the PWRZI documentation in this manual.
Entry Points
ISOSRF, EZISOS, SET3D, TRN32I, ZEROSC, STCNTR, DRCNTR, TR32, FRSTS, KURV1S, KURV2S, FRSTC, FILLIN, DRAWI, ISOSRB, MMASK

Common Blocks
ISOSR1, ISOSR2, ISOSR3, ISOSR4, ISOSR5, ISOSR6, ISOSR7, ISOSR8, ISOSR9, TEMPR, PWRZ11

Required Library Routines
The ERPRT77 package and the SPPS.

Required GKS Level
0A

I/O
Plots surface

Precision
Single

Language
FORTRAN 77

History
Developed for users of ISOSRFHR with smaller arrays.

Algorithm
Cuts through the three-dimensional array are contoured with a smoothing contourer that also marks a model of the plotting plane. Interiors of boundaries are filled in and the result is .OR.ed into another model of the plotting plane that is used to test subsequent contour lines for visibility.

Timing
Varies widely with size of T and the volume of the space enclosed by the surface drawn.

Note
Space requirements can be reduced by changing the size of the arrays ISCR, ISCA (found in COMMON ISOSR2), MASK(found in COMMON ISOSR5), and the variable NBPW (COMMON ISOSR5). ISCR and ISCA need 128x128 bits. So on a 64 bit machine ISCR, ISCA can be dimensioned to (2,128). NBPW set in subroutine MMASK should contain the number of bits per word you wish to utilize. The dimension of MASK and NMASK should equal the value of NBPW. LS should be set to the first dimension of ISCA and ISCR.

Examples
On a 60 bit machine:

    DIMENSION ISCA(4,128), ISCR(4,128)
    DIMENSION MASK(32)
    NBPW = 32

On a 64 bit machine:

    DIMENSION ISCA(2,128), ISCR(2,128)
    DIMENSION MASK(64)
    NBPW = 64
<table>
<thead>
<tr>
<th>Internal Parameters</th>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IREF</td>
<td>1</td>
<td></td>
<td>Flag to control drawing of axes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• IREF=nonzero Draw axes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• IREF=zero Do not draw axes.</td>
</tr>
</tbody>
</table>
SUBROUTINE INIT3D (EYE,NU,NV,NW,ST1,LX,NY,IS2,IU,S)

ISO-surface package for high resolution 3-dimensional arrays

<table>
<thead>
<tr>
<th>Dimension of Arguments</th>
<th>EYE(3),ST1(NV,NW,2),IS2(LX,NY),S(4), IOBJS(MV,NW)</th>
</tr>
</thead>
</table>

Latest Revision August 1987

Purpose

This package of three routines produces a perspective picture of an arbitrary object or group of objects with the hidden parts not drawn. The objects are assumed to be stored in the format described below, a format that was chosen to facilitate the display of functions of three variables or output from high resolution three-dimensional computer simulations.

Usage

The object is defined by setting elements in a three-dimensional array to one where the object is and zero where it is not. That is, the position in the array corresponds to a position in three-space, and the value of the array tells whether any object is present at that position or not. Because a large array is needed to define objects with good resolution, only a part of the array is passed to the package with each call.

There are two subroutines in the package. INIT3D is called at the beginning of a picture. This call can be skipped sometimes if certain criteria are met and certain precautions are taken. See the TIMING section for details. DANDR (Draw AND Remember) is called successively to process different parts of the three-dimensional array. For example, one plane would be processed in the first call to DANDR, while a different plane would be processed in a subsequent call. An example follows:

```
CALL INIT3D(EYE, NU, NV, NW, ST1, LX, NY, IS2, IU, S)
DO 1 IBKWDS = 1, NU
    I = NU+1-IBKWDS
    C FORM OR READ SLAB I OF THE 3 DIMENSIONAL ARRAY
    C ONLY 1 OR 0 IN THE SLAB, CALLED IOBJS
  1    CALL DANDR(NV, NW, ST1, LX, NX, NY, IS2, IU, S, IOBJS, MV)
```

ARGUMENTS

On Input

EYE An array, dimensioned 3, containing the U, V, and W coordinates of the eye position. Objects are considered to be in a box with 2 extreme corners at (1,1,1) and (NU,NV,NW). The eye position must have positive coordinates away from the coordinate planes U = 0, V = 0, and W = 0. While gaining experience with the package, use EYE(1) = 5*NU, EYE(2) = 4*NV, EYE(3) = 3*NW.

NU U direction length of the box containing the objects.

NV V direction length of the box containing the objects.
NW  W direction length of the box containing the objects.

ST1  A scratch array at least NV*NW*2 words long.

LX  The number of words needed to hold NX bits. Also, the first dimension of IS2. See NX and IS2. On a 60 bit machine, \( LX = (NX-1)/60 + 1 \).

NY  Number of cells in the Y-direction of the model of the image plane. Also the second dimension of IS2.

IS2  An array holding the image plane model. It is dimensioned LX by NY. The model is NX by NY and packed densely. If hidden lines are drawn, decrease NX and NY (and LX if possible). If visible lines are left out of the picture, increase NX and NY (and LX if need be). As a guide, some examples showing successful choices are listed:

<table>
<thead>
<tr>
<th>Given</th>
<th>NU</th>
<th>NV</th>
<th>NW</th>
<th>Resulting NX</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td>60</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>75</td>
<td>75</td>
</tr>
</tbody>
</table>

IU  Unit number of scratch file for the package. ST1 will be written NU times on this file.

S  A real array, dimensioned 4, which contains the world coordinates of the area where the picture is to be drawn. That is, all plotting coordinates generated will be bounded as follows: X coordinates will be between S(1) and S(2), Y coordinates will be between S(3) and S(4). To prevent distortion, have S(2)-S(1) = S(4)-S(3).

IOBJS  A NV by NW array (with actual first dimension MV in the calling program) describing the object. If this is the Ith call to DANDR, the part of the picture at \( U = NU+1-I \) is to be processed. IOBJS defines the objects to be drawn in the following manner: \( IOBJS(J,K) = 1 \) if any object contains the point \( (NU+1-I,J,K) \) and \( IOBJS(J,K) = 0 \) otherwise.

NX  This variable is an argument to DANDR. It is the number of cells in the X-direction of a model of the image plane. A silhouette of the parts of the picture processed so far is stored in this model. Lines to be drawn are tested for visibility by examining the silhouette. Lines in the silhouette are hidden. Lines out of the silhouette are visible. The solution is approximate because the silhouette is not formed exactly. See IS2.

MV  Actual first dimension of IOBJS in the calling program. When plotting all of IOBJS, NV = MV.
On Output

EYE, NU, NV, NW, LX, NX, NY, IU, S, IOBJS, and MV are unchanged.
ST1 and IS2 have been written in.

Note

This routine is for large arrays, 40 x 40 x 30 is a practical minimum.

Entry Points

INIT3D, PERSPC, and DANDR

Common Blocks

None

I/O

Plots visible surfaces, uses scratch file or tape.

Precision

Single

Language

FORTRAN 77

History

Originally developed at NCAR starting in late 1970.

Algorithm

The basic method is to contour cuts through the array starting with a cut nearest the observer. The algorithm leaves out the hidden parts of the contours by suppressing those lines enclosed within the lines produced while processing the preceding cuts. The technique is described in detail in the reference cited below.

Required GKS Level

0A

Reference


Accuracy

The algorithm is not exact. However, reasonable pictures are produced.

Timing

This routine is very time consuming. If many pictures are produced with the same size arrays and eye position, much time can be saved by rewinding unit IU, filling IS2 with zeros, and skipping the call to INIT3D for other than the first picture.

Portability

Two machine-dependent constants are initialized in DANDR. INIT3D has an entry statement for PERSPC. In DANDR, .AND. and .OR. are used for masking operations.
PWRITX

SUBROUTINE PWRITX (X,Y,IDPC,NCHAR,JSIZE,JOR,JCTR)

Latest Revision August 1987

Purpose PWRITX is a character plotting routine. It produces high quality characters for annotating graphs, making movie titles, etc.

Usage CALL PWRITX (X,Y,IDPC,NCHAR,JSIZE,JOR,JCTR)

ARGUMENTS

Common Block PUSER

By default, PWRITX uses the complex character set. A complete list of the characters contained in this font appears as the output from the PWRITX demo driver in the "Examples" section of this manual. PWRITX can be made to access one other font, called the duplex character set. The characters in the duplex font are somewhat simpler in appearance (no serifs, fewer curves, etc.), but are still of high-quality. Only the alphanumeric characters in the duplex font are different from the alphanumeric characters in the complex font. All other characters remain the same. To make PWRITX access the duplex character set, the user has to define in his main program a common block PUSER containing 1 integer variable. If this variable is initialized to 1 before the first call to PWRITX, the duplex character set is used by PWRITX.

For example:

. .

COMMON /PUSER/ MODE

. .

MODE = 1

. .

CALL PWRITX(X,Y,IDPC,NCHAR,JSIZE,JOR,JCTR)

. .

Note: The character set cannot be changed after the first call to PWRITX. To produce examples of the duplex character set, run the
demonstration driver for PWRITX (as supplied) with MODE set to 1 as described above. The demo driver for PWRITX also serves for examples of PWRITX calls (particularly the final plot.)

On Input  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X,Y</td>
<td>Positioning coordinates for the characters to be drawn. These are world coordinates. See JCTR.</td>
</tr>
<tr>
<td>IDPC</td>
<td>Characters to be drawn and FUNCTION CODES (see below.) IDPC is TYPE CHARACTER.</td>
</tr>
<tr>
<td>NCHAR</td>
<td>The number of characters in IDPC, including characters to be drawn and function codes.</td>
</tr>
<tr>
<td>JSIZE</td>
<td>Size of the character.</td>
</tr>
<tr>
<td></td>
<td>- If between 0 and 3, it is 1., 1.5, 2., or 3. times digitized character width. (See &quot;Function Codes&quot; below for these sizes.)</td>
</tr>
<tr>
<td></td>
<td>- If greater than 3, it is the desired plotter address units for principal character height, such as principal characters will be JSIZE plotter address units high, and indexical and cartographic characters will be scaled proportionally, such that Indexical = (13/21)*JSIZE PLA units Cartographic = (9/21)*JSIZE PLA units high</td>
</tr>
<tr>
<td>JOR</td>
<td>Character string orientation in degrees counterclockwise from the positive X axis.</td>
</tr>
<tr>
<td>JCTR</td>
<td>Centering option.</td>
</tr>
<tr>
<td></td>
<td>= 0 (X,Y) is the center of the entire string.</td>
</tr>
<tr>
<td></td>
<td>= -1 (X,Y) is the center of the left edge of the first character.</td>
</tr>
<tr>
<td></td>
<td>= 1 (X,Y) is the center of the right edge of the last character.</td>
</tr>
</tbody>
</table>

On Output  

All arguments are unchanged.

Function Codes  

Function codes may be included in the character string IDPC to change font, case, etc. within a string of text. All function instructions must be enclosed in apostrophes. No punctuation is needed between functions except for a comma between adjacent numbers; however, commas may be used between functions to improve readability. The following are the only legal function codes. Any other characters in a function string will be ignored except that an error message will be printed and, if more than 10 errors occur within a string, control will be returned to the main program. At the first call to PWRITX, size, type and case are Principal, Roman, and Upper.

PLA = plotter address (for resolution 10)  
UPA = user plotter address (resolution as defined by user)

Font Definitions  

R Roman type characters  
G Greek type characters
Size Definitions
P  Principal size, digitized to be 21 PLA units high. The total character including white space is 32 PLA units high. A carriage return or a Y increment will space down 32 PLA units. A blank or an X increment will space across 16 PLA units. Note: Characters vary in width.
I  Indexical size, digitized to be 13 PLA units high (20 PLA units high including white space). A carriage return or a Y increment is 20 PLA units. Blanks or X increments are 12 PLA units.
K  Cartographic size, digitized to be 9 PLA units high (14 PLA units high including white space). Carriage return or Y increments are 14 PLA units. Blanks or X increments are 8 PLA units.

Case Definitions
U or Un  Upper case
  If U is followed by a number n (not separated by a comma) then n characters will be drawn in upper case, subsequent characters will be in lower case. (The Un option is particularly useful for capitalizing sentences.)
L or Ln  Lower case
  If L is followed by a number n, then n characters will be drawn in lower case and subsequent characters will be in upper case.

Level Definitions
S or Sn  Superscript level.
B or Bn  Subscript level.
N or Nn  Normal level.

When super or subscripting, the character size will change depending on the previous character drawn. Principal base characters will be subscripted or superscripted with indexical characters, with a 10 PLA unit shift (scaled to JSIZE) up or down. Indexical and cartographic base characters will be sub or superscripted with cartographic characters with a 7 PLA unit shift.

The case of the indexing characters will generally be the same as that of the base character unless otherwise specified. Exception: a lower case indexical base will be super or subscripted with upper case cartographic, as the cartographic type has no lower case alphabetic or numeric characters available.

If S, B, or N is followed by a number n, then n characters will be drawn as specified above, after which character size, case, and position will be reset to that of the base character. If n is negative, its absolute value will be used instead (n cannot be 0.)

Do not overlap level definitions given for a specified number of characters. The N option returns character case and size to that of the base but maintains the current character position.
Example: 'U1'T'S1'EST
    E
Will be written TST
'U1'T'S'E'N'ST
    E
Will be written T ST

Coordinate Definitions (Descriptions assume normal UPA unit space.)
H,Hn,HnQ. Increment in the X direction.
If this option appears without a number n, n will be taken to be 1. Hn will shift the present X position n UPA units. If n is positive, the shift is to the right; if n is negative the shift is to the left. If Hn is followed by a Q, the X position will be incremented by n character widths (i.e., n blanks) either right or left.
V,Vn,VnQ. Increment in the Y direction.
If this option appears without a number n, n will be taken to be 1. Vn will shift the present Y position n UPA units. If n is positive the shift is up, if n is negative the shift is down. If Vn is followed by a Q, the Y position will be incremented by n lines up or down.
X,Xn. Set X.
If X appears without a number n, this will act as a do-nothing statement. Otherwise, the character position in the X direction will be set to the UPA coordinate n, so that the next character drawn will be centered on n and subsequent characters will be drawn from this position.
Y,Yn. Set Y.
This works the same as set X.
C Carriage return.
A carriage return will be done before the next character is plotted.

Direction Definitions
D,Dn. Write down, rather than across the frame.
If D appears without an n or if n=0, all characters will be written down, until an 'A' function is encountered. If D is followed by a number n, n characters will be written down and subsequent characters will be written across the frame. If n is negative, the absolute value of n is used instead.
A Write across.
Escape from the D option.

Direct Character Access
NNN. Where NNN stands for a numeric character.
Character number NNN will be drawn.
NNN is base 8.

Note
- All characters in a given call are drawn in the same intensity. If JSIZE .LE. 2, characters are in low intensity, otherwise they are in high intensity. Return to the main program is always in high intensity.
- On other than the first entry to PWRITX, font, case, etc. are in the state they were in when the program exited from the the last PWRITX call.
PWRITX

* Font, size, and case are reset to previous values if they were only set for a specified number of characters in the previous call.
* The previous case is always reset to upper for each new call to PWRITX.
* The direction is always reset to across for each new call to PWRITX.
* Numbers for direct character access must not be signed. All other numbers can be signed.

Portability FORTRAN 77

Required SQRT, SIN, COS.

Resident

Routines

Required GKS Level 0A

Entry Points PWRITX, GTNUM, GTNUMB, XTCH, PWRXBD, GTSIGN, GTDGTS, HTABLE, GETHOL, CCHECK, DCECK, PWRX MKMSK

Common Blocks PWRSV1, PSAV1, PSAV2, PUSER, PINIT, PINIT1, HOLTAB, PWRC0, PWRC1, PWRC2

Required Library The ERPRT77 package and the SPPS.

Library

I/O Plots characters.

Precision Single

Language FORTRAN 77

History

* Originally implemented to make the Hershey character sets more accessible.
* Made portable in 1978 for use on all computer systems which support plotters with up to 15 bit resolution.
* Made to be FORTRAN 77 and GKS compatible, August 1984.

Internal Parameters All values below are for plotter address units.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPRIH</td>
<td>32.</td>
<td>Height of principal characters.</td>
</tr>
<tr>
<td>SPRIW</td>
<td>16.</td>
<td>Width of principal characters.</td>
</tr>
<tr>
<td>SINDH</td>
<td>20.</td>
<td>Height of indexical characters.</td>
</tr>
<tr>
<td>SINDW</td>
<td>12.</td>
<td>Width of indexical characters.</td>
</tr>
<tr>
<td>SCARH</td>
<td>14.</td>
<td>Height of cartographic characters.</td>
</tr>
<tr>
<td>SCARW</td>
<td>8.</td>
<td>Width of cartographic characters.</td>
</tr>
</tbody>
</table>

August 1987
SSPR 10. Shift in case of super or subscripting for principal characters.

SSIC 7. Shift in case of super or subscripting for indexical characters.
PWRITY
PWRITY

SUBROUTINE PWRITY (X,Y,ID,N,ISIZE,ITHETA,ICNT)

Latest Revision August 1987

Purpose PWRITY is a character-plotting routine. It has some features not found in WTSTR, but is not as fancy as PWRITX.

Usage CALL PWRITY(X,Y,ID,N,ISIZE,ITHETA,ICNT)

ARGUMENTS

On Input

X,Y Positioning coordinates for the characters to be drawn. X and Y are user world coordinates and are scaled according to the current normalization transformation. Also, see ICNT.

ID Character string to be drawn.

N The number of characters in ID.

ISIZE Size of the character:
  * If between 0 and 3, ISIZE is chosen as 1., 1.5, 2., or 3. times an 8 plotter address character width.
  * If greater than 3, ISIZE is the character width in plotter address units.

ITHETA Angle, in degrees, at which the characters are plotted (counterclockwise from the positive X axis.)

ICNT Centering option:
  = -1 (X,Y) is the center of the left edge of the first character.
  = 0 (X,Y) is the center of the entire string.
  = 1 (X,Y) is the center of the right edge of the last character.

On Output All arguments are unchanged.

Common Blocks PWRCOM

Required Library SPPS

Routines

Required GKS Level 0A
<table>
<thead>
<tr>
<th><strong>I/O</strong></th>
<th>Plots characters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>Single</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>FORTRAN 77</td>
</tr>
<tr>
<td><strong>History</strong></td>
<td>Implemented for use in DASHCHAR. Made portable in January 1977 for use on computer systems that support plotters with up to 15 bits resolution. Converted to FORTRAN 77 and GKS in July 1984.</td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td>Digitizations of the characters are stored internally and adjusted according to X, Y, ISIZE and ICNT, then plotted.</td>
</tr>
<tr>
<td><strong>Timing</strong></td>
<td>Slower than WTSTR, faster than PWRITX.</td>
</tr>
<tr>
<td><strong>Portability</strong></td>
<td>FORTRAN 77</td>
</tr>
</tbody>
</table>
PWRZI
PWRZI

SUBROUTINE PWRZI (X,Y,Z,ID,N,ISIZE,LINE,ITOP,ICNT)

Latest Revision August 1987

Purpose PWRZI is a character-plotting routine for plotting characters in three-space when using ISOSRF. For a large class of possible positions, the hidden character problem is solved.

PWRZI will not work with ISOSRFHR.

Usage CALL PWRZI (X,Y,Z,ID,N,ISIZE,LINE,ITOP,ICNT)

CALL PWRZI after calling ISOSRF and before calling FRAME.

ARGUMENTS

On Input X,Y,Z Positioning coordinates for the characters to be drawn. These are floating point numbers in the same three-space as used in ISOSRF.

ID Character string to be drawn. ID is of type CHARACTER.

N The number of characters in ID.

ISIZE Size of the character:

- If between 0 and 3, ISIZE is 1, 1.5, 2, or 3 times a standard width equal to 1/128th of the screen width.
- If greater than 3, ISIZE is the character width in plotter address units.

LINE The direction in which the characters are to be written.

1 = +X   -1 = -X
2 = +Y   -2 = -Y
3 = +Z   -3 = -Z

ITOP The direction from the center of the first character to the top of the first character (the potential values for ITOP are the same as those for LINE as given above). Note that LINE cannot equal ITOP even in absolute value.

ICNT Centering option.

-1 (X,Y,Z) is the center of the left edge of the first character.
0 (X,Y,Z) is the center of the entire string.
1 (X,Y,Z) is the center of the right edge of the last character.

On Output All arguments are unchanged.
Note  The hidden character problem is solved correctly for characters near (but not inside) the three-space object.

Entry Points  PWRZI, INITZI, PWRZ0I, PWRZGI

Common Blocks  PWRZ1I,PWRZ2I

I/O  Plots character(s)

Precision  Single

Required Library Routines  ISOSRF, the ERPRT77 package, and the SPPS OA

Required GKS Level  0A

Language  FORTRAN 77

History  Implemented for use with ISOSRF
PWRZS
PWRZS

SUBROUTINE PWRZS (X,Y,Z,ID,N,ISIZE,LINE,ITOP,ICNT)

Latest Revision August 1987

Purpose PWRZS is a character-plotting routine for plotting characters in three-space when using SRFACE. For a large class of possible positions, the hidden character problem is solved.

Usage CALL PWRZS (X,Y,Z,ID,N,ISIZE,LINE,ITOP,ICNT)
Use CALL PWRZS after calling SRFACE and before calling FRAME. Note: SRFACE will have to be changed to suppress the FRAME call. See IFR in SRFACE internal parameters.

ARGUMENTS

On Input X,Y,Z Positioning coordinates for the characters to be drawn. These are floating point numbers in the same three-space as used in SRFACE.
ID Character string to be drawn. ID is of type CHARACTER.
N The number of characters in ID.
ISIZE Size of the character:
  • If between 0 and 3, ISIZE is 1., 1.5, 2., or 3. times a standard width equal to 1/128th of the screen width.
  • If greater than 3, ISIZE is the character width in plotter address units.
LINE The direction in which the characters are to be written.
  1 = +X  -1 = -X
  2 = +Y  -2 = -Y
  3 = +Z  -3 = -Z
ITOP The direction from the center of the first character to the top of the first character (the potential values for ITOP are the same as those for LINE as given above.) Note that LINE cannot equal ITOP even in absolute value.
ICNT Centering option.
  -1 (X,Y,Z) is the center of the left edge of the first character.
  0 (X,Y,Z) is the center of the entire string.
  1 (X,Y,Z) is the center of the right edge of the last character.

On Output All arguments are unchanged.

Note The hidden character problem is solved correctly for characters near (but not inside) the three-space object.
Entry Points  PWRZS, INITZS, PWRZOS, PWRZGS
Common Blocks  PWRZ1S,PWRZ2S
I/O  Plots character(s)
Precision  Single
Required Library Routines  SRFACE, the ERPRT77 package, and the SPPS
Required GKS Level  0A
Language  FORTRAN 77
History  Implemented for use with SRFACE.
PWRZT
PWRZT

SUBROUTINE PWRZT (X,Y,Z,ID,N,ISIZE,LINETHREE,ITOP,ICNT)

Latest Revision August 1987.

Purpose PWRZT is a character-plotting routine for plotting characters in three-
space when using THREED. For a large class of possible positions, the
hidden character problem is solved.

Usage CALL PWRZT (X,Y,Z,ID,N,ISIZE,LINETHREE,ITOP,ICNT)
Use CALL PWRZT after calling
THREED and before calling FRAME.

ARGUMENTS

On Input X,Y,Z Positioning coordinates for the characters to be drawn. These
are floating point numbers in the same three-space as used in
THREED.

ID Character string to be drawn. ID is of type CHARACTER.

N The number of characters in ID.

ISIZE Size of the character:
- If between 0 and 3, ISIZE is 1., 1.5, 2., or 3. times a standard
  width equal to 1/128th of the screen width.
- If greater than 3, ISIZE is the character width in plotter ad-
  dress units.

LINE The direction in which the characters are to be written.
1 = +X  -1 = -X
2 = +Y  -2 = -Y
3 = +Z  -3 = -Z

ITOP The direction from the center of the first character to the top of
the first character (the potential values for ITOP are the same
as those for LINE as given above.) Note that LINE cannot equal
ITOP even in absolute value.

ICNT Centering option.
-1 (X,Y,Z) is the center of the left edge of the first character.
0 (X,Y,Z) is the center of the entire string.
1 (X,Y,Z) is the center of the right edge of the last character.

On Output All arguments are unchanged.

Note The hidden character problem is solved correctly for characters near (but
not inside) the three-space object.
Entry Points  PWRZT, INITZT, PWRZOT, PWRZGT

Common Blocks  PWRZ1T, PWRZ2T

I/O  Plots character(s)

Precision  Single

Required Library Routines  THREED, the ERPRT77 package, and SPPS

Required GKS Level  0A

Language  FORTRAN 77

History  Implemented for use with THREED
SRFACE
SRFACE

SUBROUTINE SRFACE (X,Y,Z,M,MX,NX,NY,S,STEREO)

Dimension of Arguments

X(NX), Y(NY), Z(MX,NY), M(2,NX,NY), S(6)

Latest Revision

August 1987

Purpose

SRFACE draws a perspective picture of a function of two variables with hidden lines removed. The function is approximated by a two-dimensional array of heights.

Usage

If the following assumptions are met, use

CALL EZSRFC (Z,M,N,ANGH,ANGV,WORK)

Assumptions:

- the entire array is to be drawn,
- the data are equally spaced (in the X-Y plane),
- no stereo pairs,
- scaling is chosen internally.

If these assumptions are not met, use

CALL SRFACE (X,Y,Z,M,MX,NX,NY,S,STEREO)

ARGUMENTS

On Input for EZSRFC

Z The M by N array to be drawn.
M The first dimension of Z.
N The second dimension of Z.
ANGH Angle in degrees in the X-Y plane to the line of sight (counterclockwise from the plus-X axis).
ANGV Angle in degrees from the X-Y plane to the line of sight (positive angles are above the middle Z, negative below).
WORK A scratch storage dimensioned at least 2*M*N+M+N.

On Output for EZSRFC

Z, M, N, ANGH, ANGV are unchanged. WORK has been written in.
ARGUMENTS

On Input for SRFACE

X  A linear array NX long containing the X coordinates of the points in the surface approximation. See the "Notes" below.

Y  The linear array NY long containing the Y coordinates of the points in the surface approximation. See the "Notes" below.

Z  An array MX by NY containing the surface to be drawn in NX by NY cells. $Z(I,J) = F(X(I),Y(J))$. See the "Notes" below.

M  Scratch array at least 2*NX*NY words long.

MX  First dimension of Z.

NX  Number of data values in the X direction (the first subscript direction) in Z to be plotted. When plotting an entire array, MX=NX.

NY  Number of data values in the Y direction (the second subscript direction) to be plotted.

S  S defines the line of sight. The viewer's eye is at $(S(1), S(2), S(3))$ and the point looked at is at $(S(4), S(5), S(6))$. The eye should be outside the block with opposite corners $(X(1), Y(1), ZMIN)$ and $(X(NX), Y(NY), ZMAX)$ and the point looked at should be inside it. For a nice perspective effect, the distance between the eye and the point looked at should be 5 to 10 times the size of the block. See the "Notes" below.

STEREO  Flag to indicate if stereo pairs are to be drawn. 0.0 means no stereo pair (one picture). Non-zero means put out two pictures. The value of STEREO is the relative angle between the eyes. A value of 1.0 produces standard separation. Negative STEREO reverses the left and right figures. See the documentation below for internal variable ISTP for additional information.

On Output for SRFACE

X, Y, Z, MX, NX, NY, S, STEREO are unchanged. M has been written in.

Notes

- The range of Z compared with the range of X and Y determines the shape of the picture. They are assumed to be in the same units and not wildly different in magnitude. S is assumed to be in the same units as X, Y, and Z.

- Picture size can be made relative to distance. See comments in SETR.

- TRN32S can be used to translate from 3 space to 2 space. See comments there.

- Data with extreme discontinuities may cause visibility errors. If this problem occurs, use a distant eye position away from the +Z axis.
- The default line color is set to color index 1. If the user wishes to change the line color, he can do so by defining color index 1 before calling SRFACE, or by putting the common block SRFINT in his calling program and defining and using color index ISRFMJ (defaulted to 1 in BLOCKDATA).

<table>
<thead>
<tr>
<th>Entry Points</th>
<th>SRFACE, SRFGK, EZSRFC, SETR, DRAWS, TRN32S, CLSET, CTCCELL, SRFABD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Blocks</td>
<td>SRFACE, SRFINT, SRFBLK, PWRZIS, SRFIP1</td>
</tr>
<tr>
<td>Required Library</td>
<td>The SPPS</td>
</tr>
<tr>
<td>Required GKS Level</td>
<td>0A</td>
</tr>
<tr>
<td>I/O</td>
<td>Plots</td>
</tr>
<tr>
<td>Precision</td>
<td>Single</td>
</tr>
<tr>
<td>Language</td>
<td>FORTRAN 77</td>
</tr>
</tbody>
</table>
| History | Converted to FORTRAN 77 and GKS in March 1984.  
Prepared for SIGGRAPH, August 1976.  
Written in December 1971. Replaced K.S.+G. algorithm called SOLIDS at NCAR.  |
| Algorithm | The data are processed from the near side of the surface to the far side.  
Visibility information is stored (see reference). Highest so far is visible from above.  |
| Accuracy | If the ends of a line segment are visible, the middle is assumed visible.  |
| Timing | Proportional to NX*NY.  |

August 1987
### Internal Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
</table>
| IFR  | 1       | -1 Call FRAME first.  
|      |         | 0 Do not call FRAME.  
|      |         | +1 Call FRAME when done.  |
| ISTP | 0       | STEREO type if STEREO non-zero.  
|      |         | -1 Alternating frames, slightly offset (for movies, IROTS = 0).  
|      |         | 0 Blank frame between for stereo slide.  
|      |         | IROTS = 1).  |
| IROTS| 0       | 0 +Z in vertical plotting direction (CINE mode).  
|      |         | +1 +Z in horizontal plotting direction (COMIC mode).  |
| IDRX | 1       | +1 Draw lines of constant X.  
|      |         | 0 Do not.  |
| IDRY | 1       | +1 Draw lines of constant Y.  
|      |         | 0 Do not.  |
| IDRZ | 0       | +1 Draw lines of constant Z (contour lines).  
|      |         | 0 Do not.  |
| IUPPER| 0       | +1 Draw upper side of surface.  
|      |         | 0 Draw both sides.  
|      |         | -1 Draw lower side.  |
| ISKIRT| 0       | +1 Draw a skirt around the surface.  
|      |         | BOTTOM = HSKIRT.  
|      |         | 0 Do not.  |
| NCLA | 6       | Approximate number of levels of constant Z that are drawn if levels are not specified. 40 levels maximum.  |
| THETA| .02     | Angle, in radians, between eyes for stereo pairs.  |
| HSKIRT| 0       | Height of skirt (if ISKIRT = 1).  |
| CHI  | 0       | Highest level of constant Z.  |
| CLO  | 0       | Lowest level of constant Z.  |
| CINC | 0       | Increment between levels.  |

(If CHI, CLO, or CINC is zero, a nice value is generated automatically.)
<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOFFP</td>
<td>0</td>
<td>Flag to control use of special value feature. Do not have both IOFFP=1 and ISKIRT=1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0  Feature not in use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1  Feature in use. No lines drawn to data points in Z that are equal to SPVAL.</td>
</tr>
<tr>
<td>SPVAL</td>
<td>0.</td>
<td>Special value used to mark unknown data when IOFFP=1.</td>
</tr>
</tbody>
</table>
STRMLN

SUBROUTINE STRMLN (U,V,WORK,IMAX,IPTSX,JPTSY,NSET,IER)

Dimension of Arguments
U(IMAX,JPTSY), V(IMAX,JPTSY), WORK(2*IMAX*JPTSY)

Latest Revision
August 1987

Purpose
STRMLN draws a streamline representation of the flow field. The representation is independent of the flow speed.

Usage
If the following assumptions are met, use

CALL EZSTRM (U,V,WORK,IMAX,JMAX)

Assumptions:
- The whole array is to be processed.
- The arrays are dimensioned U(IMAX,JMAX), V(IMAX,JMAX), and WORK(2*IMAX*JMAX).
- Window and viewport are to be chosen by STRMLN.
- PERIM is to be called.

If these assumptions are not met, use

CALL STRMLN (U,V,WORK,IMAX,IPTSX,JPTSY,NSET,IER)

The user must call FRAME in the calling routine.

The user may change various internal parameters via common blocks. See below.

ARGUMENTS

On Input  U, V    Two dimensional arrays containing the velocity fields to be plotted.

Note: If the U and V components are, for example, defined in Cartesian coordinates and the user wishes to plot them on a different projection (such as stereographic), then the appropriate transformation must be made to the U and V components via the functions FU and FV (located in DRWSTR).

WORK User provided work array. The dimension of this array must be .GE. 2*IMAX*JPTSY.

Caution: This routine does not check the size of the work array.

IMAX  The first dimension of U and V in the calling program (X-direction).
**IPTSX** The number of points to be plotted in the first subscript direction (X-direction).

**JPTS Y** The number of points to be plotted in the second subscript direction (Y-direction).

**NSET** Flag to control scaling
- $>0$ STRMLN assumes that the window and viewport have been set by the user in such a way as to properly scale the plotting instructions generated by STRMLN. PERIM is not called.
- $=0$ STRMLN will establish the window and viewport to properly scale the plotting instructions to the standard configuration. PERIM is called to draw the border.
- $<0$ STRMLN establishes the window and viewport so as to place the streamlines within the limits of the user’s window. PERIM is not called.

**On Output** Only the IER argument may be changed. All other arguments are unchanged.

**IER**
- $=0$ when no errors are detected
- $=-1$ when the routine is called with ICYC, .NE. 0 and the data are not cyclic (ICYC is an internal parameter described below); in this case the routine will draw the streamlines with the non-cyclic interpolation formulas.

**Entry Points** STRMLN, DRWSTR, EZSTRM, GNEWPT, CHKCYC

**Common Blocks** STR01, STR02, STR03, STR04

**Required Library Routines** GRIDAL, GBYTES, and the SPPS

**Required GKS Level** 0A

**I/O** None

**Precision** Single

**Language** FORTRAN 77


**Portability** FORTRAN 77

**Algorithm** Wind components are normalized to the value of DISPL. The least significant two bits of the work array are utilized as flags for each grid box. Flag 1 indicates whether any streamline has previously passed through this box. Flag 2 indicates whether a directional arrow has already appeared in a box. Judicious use of these flags prevents overcrowding of streamlines and directional arrows. Experience indicates that a
final pleasing picture is produced when streamlines are initiated in the center of a grid box. The streamlines are drawn in one direction then in the opposite direction.

**Reference**

The techniques utilized here are described in an article by Thomas Whitaker (U. of Wisconsin), which appeared in the notes and correspondence section of *Monthly Weather Review*, June 1977.

**Timing**

Highly variable. It depends on the complexity of the flow field and the parameters: DISPL, DISPC, CSTOP, INITA, INITB, ITERC, and IGFLG. (See below for a discussion of these parameters.) If all values are default, then a simple linear flow field for a 40 x 40 grid will take about 0.4 seconds on the CRAY1-A; a fairly complex flow field will take about 1.5 seconds on the CRAY1-A.

<table>
<thead>
<tr>
<th>Internal Parameters</th>
<th>Name</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXT</td>
<td>0.25</td>
<td>Lengths of the sides of the plot are proportional to IPTSX and JPTSY except in the case when MIN(IPTSX,JPT) / MAX(IPTSX,JPTSY) .LT. EXT; in that case, a square graph is plotted.</td>
</tr>
<tr>
<td></td>
<td>SIDE</td>
<td>0.90</td>
<td>Length of longer edge of plot. (See also EXT.)</td>
</tr>
<tr>
<td></td>
<td>XLT</td>
<td>0.05</td>
<td>Left hand edge of the plot. (0.0 = left edge of frame)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1.0 = right edge of frame)</td>
</tr>
<tr>
<td></td>
<td>YBT</td>
<td>0.05</td>
<td>Bottom edge of the plot. (0.0 = bottom; 1.0 = top)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(YBT+SIDE and XLT+SIDE must be .LE. 1.)</td>
</tr>
<tr>
<td></td>
<td>INITA</td>
<td>2</td>
<td>Used to precondition grid boxes to be eligible to start a streamline. For example, a value of 4 means that every fourth grid box is eligible; a value of 2 means that every other grid box is eligible (see INITB).</td>
</tr>
<tr>
<td></td>
<td>INITB</td>
<td>2</td>
<td>Used to precondition grid boxes to be eligible for direction arrows. If the user changes the default values of INITA and/or INITB, it should be done such that MOD(INITA,INITB) = 0. For a dense grid try INITA=4 and INITB=2 to reduce the CPU time.</td>
</tr>
<tr>
<td></td>
<td>AROWL</td>
<td>0.33</td>
<td>Length of direction arrow. For example, 0.33 means each directional arrow will take up a third of a grid box.</td>
</tr>
<tr>
<td>Name</td>
<td>Default</td>
<td>Function</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>ITERP</td>
<td>35</td>
<td>Every 'ITERP' iterations the streamline progress is checked.</td>
<td></td>
</tr>
<tr>
<td>ITERC</td>
<td>-99</td>
<td>The default value of this parameter is such that it has no effect on the code. When set to some positive value, the program will check for streamline crossover every 'ITERC' iterations. (The routine currently does this every time it enters a new grid box.)</td>
<td></td>
</tr>
<tr>
<td>Caution: When this parameter is activated CPU time will increase.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IGFLG</td>
<td>0</td>
<td>A value of zero means that the sixteen point Bessel Interpolation Formula will be utilized where possible; when near the grid edges, quadratic and bi-linear interpolation will be used. This mixing of interpolation schemes can sometimes cause slight raggedness near the edges of the plot. If IGFLG.NE.0, then only the bilinear interpolation formula is used; this will generally result in slightly faster plot times but a less pleasing plot.</td>
<td></td>
</tr>
<tr>
<td>IMSG</td>
<td>0</td>
<td>If zero, then no missing U and V components are present. If .NE. 0, STRMLN will utilize the bi-linear interpolation scheme and terminate if any data points are missing.</td>
<td></td>
</tr>
<tr>
<td>UVMSG</td>
<td>1.E+36</td>
<td>Value assigned to a missing point.</td>
<td></td>
</tr>
<tr>
<td>ICYC</td>
<td>0</td>
<td>Zero means the data are non-cyclic in the X direction. If .NE 0, the cyclic interpolation formulas will be used.</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td></td>
<td>Even if the data are cyclic in X, leaving ICYC = 0 will do no harm.</td>
<td></td>
</tr>
<tr>
<td>DISPL</td>
<td>0.33</td>
<td>The wind speed is normalized to this value. (See the discussion below.)</td>
<td></td>
</tr>
<tr>
<td>DISPC</td>
<td>0.67</td>
<td>The critical displacement. If after 'ITERP' iterations the streamline has not moved this distance, the streamline will be terminated.</td>
<td></td>
</tr>
<tr>
<td>CSTOP</td>
<td>0.50</td>
<td>This parameter controls the spacing between streamlines. The checking is done when a new grid box is entered.</td>
<td></td>
</tr>
</tbody>
</table>
Discussion of DISPL, DISPC, and CSTOP

Assume a value of 0.33 for DISPL. This means that it will take three steps to move across one grid box if the flow was all in the X direction. If the flow is zonal, then a larger value of DISPL is in order. If the flow is highly turbulent, then a smaller value is in order.

Note: The smaller DISPL, the more the CPU time. A value of 2 to 4 times DISPL is a reasonable value for DISPC. DISPC should always be greater than DISPL. A value of 0.33 for CSTOP would mean that a maximum of three stream-lines will be drawn per grid box. This max will normally only occur in areas of singular points.

Note: Any or all of the above parameters may be changed by utilizing common blocks STR02 and/or STR03.

UXSML 1.E-50 The smallest real number on the host computer. This is set automatically by R1MACH.

NCHK 750 This parameter is located in DRWSTR. It specifies the length of the circular lists used for checking for STRMLN crossovers. For most plots this number may be reduced to 500 or less and the plots will not be altered.

ISKIP Number of bits to be skipped to get to the least two significant bits in a floating point number. The default value is set to I1MACH(5) - 2. This value may have to be changed depending on the target computer; see subroutine DRWSTR.
THREED
THREED

Three-dimensional line drawing package

Latest Revision  August 1987

Purpose  THREED is a package of subroutines that provides line drawing capabilities in three-space.

Usage  Each entry point in this package is described below.

SET3 (XA,XB,YA,YB,UC,UD,VC,VD,WC,WD,EYE)

XA, XB, YA, YB define the portion of the plotting surface into which the user's plot will be placed. These values should be in the range 0. to 1. For example, if one wants the plot to occupy the maximum plotting surface, set XA=0., YA=0., XB=1., YB=1.; if one wants the plot to appear in the lower left corner of the plotting surface, set XA=0., YA=0., XB=.5, YB=.5 .

UC, UD, VC, VD, WC, and WD define a volume in user-coordinate space which will be transformed onto the plotting surface defined by XA, XB, YA, YB.

EYE is an array, 3 words long, containing the U, V, and W coordinates of the EYE position. All lines in the plot are drawn as viewed from the EYE. EYE is specified in user coordinates and should be outside the box defined by UC, UD, VC, VD, WC, and WD.

CURVE3 (U,V,W,N)

Draws a curve through N points. The points are defined by the linear arrays U, V, and W, which are dimensioned N or greater.

LINE3 (UA,VA,WA,UB,VB,WB)

Draws a line connecting the coordinates (UA,VA,WA) and (UB,VB,WB).

FRST3 (U,V,W)

Positions the pen to (U,V,W).

VECT3 (U,V,W)

Draws a line between the current pen position and the point (U,V,W). The current pen position becomes (U,V,W). Note that a curve can be drawn by using a FRST3 call followed by a sequence of VECT3 calls.
POINT3 (U,V,W)

Plots a point at (U,V,W).

PERIM3 (MAGR1,MINR1,MAGR2,MINR2,IWHICH,VAR)

Draws a perimeter with tick marks.

IWHICH designates the normal vector to the perimeter drawn (1=U, 2=V, 3=W).

VAR is the value on the axis specified by IWHICH where the perimeter is to be drawn.

MAGR1 and MAGR2 specify the number of major tick marks to be drawn in the two coordinate directions.

MINR1 and MINR2 specify the number of minor ticks between each major tick.

MAGR1, MAGR2, MINR1 and MINR2 are specified by the number of divisions (holes), not the number of ticks. So if MAGR1=1, there would be no major divisions.

TICK43 (MAGU,MINU,MAGV,MINV,MAGW,MINW)

TICK43 allows program control of tick mark length in subroutine PERIM3. MAGU, MAGV, MAGW specify the length, in plotter address units of major division tick marks on the U, V, and W axes. MINU, MINV, MINW specify the length, in plotter address units of minor division tick marks on the U, V, and W axes.

FENCE3 (U,V,W,N,IOREN,BOT)

This entry is used to draw a line in three-space as well as a "fence" between the line and a plane normal to one of the coordinate axes.

The arguments U, V, W, and N are the same as for CURVE3, described above.

IOREN specifies the direction in which the fence lines are to be drawn (1 indicates parallel to the U-axis, 2 indicates parallel to the V-axis, and 3 indicates parallel to the W-axis.)

BOT specifies where the bottom of the fence is to be drawn. If the fence lines are to be drawn parallel to the W-axis, and BOT=2., then the bottom of the fence would be the plane W=2.
Note
For drawing characters in conjunction with THREED, use the companion routine PWRZT.

Entry Points

Common Blocks
TEMPR, SET31, PWRZ1T, TCK31, PRM31, THRINT

Required Library Routines
PWRZT and the SPPS

Required GKS Level
0A

History
Written and standardized in November 1973.

I/O
Plots lines.

Precision
Single

Language
FORTRAN 77

Accuracy
+ or -.5 plotter address units per call. There is no cumulative error.

Portability
ANSI FORTRAN 77
VELVCT
VELVCT

SUBROUTINE VELVCT (U,LU,V,LV,M,N,FLO,HI,NSET,LENGTH,ISPV,SPV)

Dimension of
Arguments U(LU,N),V(LV,N),SPV(2)

Latest Revision August 1987

Purpose VELVCT draws a representation of a two-dimensional velocity field by drawing arrows from each data location. The length of the arrow is proportional to the strength of the field at that location and the direction of the arrow indicates the direction of the flow at that location.

Usage If the following assumptions are met, use

CALL EZVEC (U,V,M,N)

Assumptions:
• The whole array is processed.
• The scale factor is chosen internally.
• The perimeter is drawn.
• FRAME is called after plotting.
• There are no special values.

If these assumptions are not met, use

CALL VELVCT (U,LU,V,LV,M,N,FLO,HI,NSET,LENGTH,ISPV,SPV)

ARGUMENTS

On Input U,V The (origins of the) two-dimensional arrays containing the velocity field to be plotted. The vector at the point (I,J) has magnitude $\sqrt{U(I,J)^2 + V(I,J)^2}$ and direction $\text{ATAN2}(V(I,J),U(I,J))$. Other representations, such as $(R,\text{THETA})$, can be plotted by changing statement functions in this routine.

LU The first dimension of U in the calling program.
LV The first dimension of V in the calling program.
M The number of data values to be plotted in the X-direction (the first subscript direction). When plotting the entire array, $LU = LV = M$.
N The number of data values to be plotted in the Y-direction (the second subscript direction).
FLO The minimum vector magnitude to be shown.
HI  The maximum vector magnitude to be shown. (A value less than or equal to zero causes the maximum value of \(\text{SQRT}(U**2+V**2)\) to be used.)

NSET  Flag to control scaling -

If NSET is zero, VELVCT establishes the window and viewport to properly scale plotting instructions to the standard configuration. PERIM is called to draw a border.

If NSET is greater than zero, VELVCT assumes that the user has established the window and viewport in such a way as to properly scale the plotting instructions generated by VELVCT. PERIM is not called.

If NSET is less than zero, VELVCT places the contour plot within the limits of the user's current window and viewport. PERIM is not called.

LENGTH  The length, in Plotter Address Units (PAUs), of a vector having magnitude HI (or, if HI=0, the length in PAUs of the longest vector). If LENGTH=0, a value is chosen such that the longest vector could just reach to the tail of the next vector. If the horizontal and vertical resolutions of the plotter are different, LENGTH should be non-zero and specified as a horizontal distance.

ISPV  Flag to control the special value feature.

0 means that the feature is not in use.

1 means that if the value of \(U(I,J) = \text{SPV}(1)\) the vector will not be plotted.

2 means that if the value of \(V(I,J) = \text{SPV}(2)\) the vector will not be plotted.

3 means that if either \(U(I,J) = \text{SPV}(1)\) or \(V(I,J) = \text{SPV}(2)\) then the vector will not be plotted.

4 means that if \(U(I,J) = \text{SPV}(1)\) and \(V(I,J) = \text{SPV}(2)\), the vector will not be plotted.

SPV  An array of length 2 that gives the value in the U array and the value in the V array that denote missing values. This argument is ignored if ISPV=0.

On Output  All arguments remain unchanged.

Note  The endpoints of each arrow drawn are \((FX(X,Y), FY(X,Y))\) and \((MXF(X,Y,U,V,SFX,SYF,MX,MY))\), \(MYF(X,Y,U,V,SFX,SYF,MY,MY)\) where \(X=I\), \(Y=J\), \(U=U(I,J)\), \(V=V(I,J)\), and SFX and SFY are scale factors. Here I is the X-index and J is the Y-index. (MX,MY) is the location of the tail. Thus the actual length of the arrow is \(\text{SQRT}(DX**2+DY**2)\) and...
the direction is \( \text{ATAN2}(DX,DY) \), where \( DX=MX-MXF(\ldots) \) and \( DY=MY-MYF(\ldots) \).

**Entry Points**  
VELVCT,EZVECT,DRWVEC,VELVEC,VELDAT

**Common Blocks**  
VEC1,VEC2

**I/O**  
Plots the vector field.

**Precision**  
Single

**Language**  
FORTRAN 77

**Required Library Routines**  
GRIDAL and the SPPS

**Required GKS Level**  
0A

**History**  
Written and standardized in November 1973. Revised in May 1975, to include MXF and MYF. Revised in March 1981, to fix certain errors; to use FL2INT and PLOTIT instead of MXMY, FRSTPT, and VECTOR; and to make the arrowheads narrower. Converted to FORTRAN 77 and GKS in July 1984.

**Algorithm**  
Each vector is examined, possibly transformed, then plotted.

**Portability**  
FORTRAN 77

**Note**  
Using this routine to put vectors on an arbitrary background drawn by SUPMAP is a bit tricky. The arithmetic statement functions FX and FY are easy to replace. The problem arises in replacing MXF and MYF. The following example may be helpful. (SUPMAP is an entry point in the EZMAP package.)

Suppose that we have two arrays, CLON(36,9) and CLAT(36,9), which contain the E-W and N-S components of a wind flow field on the surface of the earth. CLON(I,J) is the magnitude of the easterly flow. CLAT(I,J) is the magnitude of the northerly flow at a longitude \((I-1) \times 10\) degrees east of Greenwich and a latitude \((J-1) \times 10\) degrees north of the equator. SUPMAP is to be used to draw a polar projection of the earth and VELVCT is to be used to superimpose vectors representing the flow field on it. The following steps would be necessary:

1. CALL SUPMAP (1,90.,0.,-90.,90.,90.,90.,90.,-4,10,0,1,IER) to draw the map.

2. CALL VELVCT (CLON,36,CLAT,36,36,9,0.,0.,1,50,0,0.) to put vectors on it. Notice that NSET has the value 1 to tell VELVCT that SUPMAP has done the required SET call.
3. In order to ensure that step 2 will work properly, delete the arithmetic statement functions FX, FY, MXF, and MYF from VELVCT and include the following functions.

```fortran
FUNCTION FX(XX,YY)
CALL MAPTRN (10.*(YY-1.),10.*(XX-1.),X,Y)
FX=X
RETURN
END

FUNCTION FY(XX,YY)
CALL MAPTRN (10.*(YY-1.),10.*(XX-1.),X,Y)
FY=Y
RETURN
END

FUNCTION MXF (XX, YY, UU, VV, SFX, SFY, MX, MY)
CFCT=COS(.17453292519943*(YY-1.))
CALL MAPTRN(10.*(YY-1.) ,10.*(XX-1.) ,X1,Y1)
CALL MAPTRN(10.*(YY-1.)+1.E-6*VV,10.*(XX-1.)+1.E-6*UU/CFCT,X2,Y2)
U=((X2-X1)/SQRT((X2-X1)**2+(Y2-Y1)**2)) * SQRT(UU**2+VV**2)
MXF=MX+IFIX(SFX*U)
RETURN
END

FUNCTION MYF (XX, YY, UU, VV, SFX, SFY, MX, MY)
CFCT=COS (.17453292519943*(YY-1.))
CALL MAPTRN(10.*(YY-1.) ,10.*(XX-1.) ,X1,Y1)
CALL MAPTRN(10.*(YY-1.)+1.E-6*VV,10.*(XX-1.)+1.E-6*UU/CFCT,X2,Y2)
V=((Y2-Y1)/SQRT((X2-X1)**2+(Y2-Y1)**2)) * SQRT(UU**2+VV**2)
MYF=MY+IFIX(SFY*V)
RETURN
END
```

The basic notion behind the coding of the MXF and MYF functions is as follows. Since UU and VV are the longitudinal and latitudinal components, respectively, of a velocity vector having units of distance over time, 1.E-6*UU/COS(latitude) and 1.E-6*VV represent the change in longitude and latitude, respectively, of a particle moving with the flow field for a very short period of time. The routine MAPTRN is used to find the position of the particle's projection at the beginning and end of that tiny time slice and, therefore, the direction in which to draw the arrow representing the velocity vector so that it will be tangent to a projected flow line of the field at that point. The values U and V are computed so as to give the arrow the length implied by UU and VV. (The code ensures that SQRT(U**2+V**2) is equal to SQRT(UU**2+VV**2).) The length of the arrow represents the magnitude of the velocity vector, unaffected by perspective. The scaling set up by VELVCT will therefore be appropriate for the arrows drawn.

This method is rather heuristic and has three inherent problems. First, the constant 1.E-6 may need to be made larger or smaller, depending on the magnitude of your U/V data. Second, the north and south poles must be avoided. At either pole, CFCT goes to zero, giving a division by zero;
in a small region near the pole, the method may try to use MAPTRN with a latitude outside the range (-90,+90). Third, the projection must be set up so as to avoid having vector base points at the exact edge of the map. Vectors there will be of the correct length, but they may be drawn in the wrong direction (when the projected particle track determining the direction crosses the edge and reappears elsewhere on the map). With a little care, the desired results may be obtained.
Section V: Plot Examples with Source Codes

How This Section Is Organized ............................................................... 401

Higher-level Utilities .............................................................................. 403
  AREAS Example .................................................................................. 405
  AUTOGRAPH Example ......................................................................... 413
  Standard Contouring .......................................................................... 421
    CONREC + DASHCHAR Example .................................................... 421
    CONRAN + DASHCHAR Example ................................................... 426
  Quick Contouring ................................................................................ 431
    CONRECQCK + DASHLINE Example .............................................. 431
    CONRAQ + DASHLINE Example ................................................... 435
  Smooth Contouring .............................................................................. 439
    CONRECSMTH + DASHSMTH Example .......................................... 439
    CONRAN + DASHSMTH Example ................................................... 443
  Super Contouring ................................................................................. 447
    CONRECSPR + DASHSUPR Example .............................................. 447
    CONRAS + DASHSUPR Example .................................................... 451
  DASHCHAR Example ............................................................................ 455
  DASHLINE Example ............................................................................ 459
  DASHSMTH Example ........................................................................... 463
  DASHSUPR Example ........................................................................... 467
  EZMAP Example .................................................................................. 471
  EZMAPA Example ................................................................................ 479
  GRIDAL Example ................................................................................ 485
  HAFTON Example ............................................................................... 497
  HISTGR Example ................................................................................ 505
  ISOSRF Example ................................................................................ 517
  ISOSRFHFR Example .......................................................................... 523
  PWRTYX Example .............................................................................. 527
  PWRTFY Example ............................................................................... 535
  PWRTZI + ISOSRF Example .............................................................. 539
  PWRTZS + SFACE Example ............................................................... 543
  PWRTZT + THREEED Example .......................................................... 547
  SFACE Example .................................................................................. 551
  STRMLN Example ............................................................................... 557
  THREEED Example ............................................................................. 561
  VEYVECT Example .............................................................................. 565

Complex Uses of Higher-level Utilities .................................................. 571
  Transforming Coordinate Output .......................................................... 573
  Frame 1: AUTOGRAPH ......................................................................... 576
  Frame 2: CONREC + FX + FY (Polar Projection) ....................................... 577
  Frame 3: CONREC + EZMAP + FX + FY (Full Globe) ............................... 578
  Frame 4: CONREC + EZMAP + FX + FY (U.S.A.) ..................................... 579
  Frame 5: CONREC + FX + FY (Rectangular Projection) ......................... 580
  Frame 6: VEYVECT + EZMAP + FX + FY ............................................... 581
Plot Examples with Source Codes

How This Section Is Organized

This section contains examples and source codes for the examples only. (For the descriptions of how to use each utility, see the "Utilities" section of this manual.) The examples are divided into the following five categories:

**Higher-level Utilities:** The test plot examples and source codes for the test drivers of the utilities are arranged in alphabetical order, with the exception of the contouring utilities, which are arranged by function. The contouring utilities are arranged in four functional groups:

- standard contouring
- quick contouring
- smooth contouring
- super contouring

Each contouring utility must be used with one of the DASH utilities.

**Important Note:** The System Plot Package Simulator (SPPS) and a GKS package must be present for these examples to run. In addition, some of the examples require other utilities, which are listed under "Required Routines" in the comments for the test driver for each utility plot source code.

**Complex Uses of Higher-level Utilities:** These are examples of more complex uses of the utilities. Codes for color examples and the code for the cover of the manual are included.

**SPPS Test:** SPPS test plot examples with source codes are in this section.

**GKS Level 0A Output Primitives:** This section contains the code and plots for five basic GKS primitives: POLYLINE, POLYMARKER, TEXT, FILL AREA, and CELL ARRAY.

**Computer Graphics Metafile (CGM) Plot with Hex and Octal Dumps:** This example is for use by the package installer. This dump can be used to see if a metafile is created correctly.
Higher-level Utilities
AREAS Example

SUBROUTINE TAREAS (IERROR)

PURPOSE
To provide a simple demonstration of the use of AREAS.

USAGE
CALL TAREAS (IERROR)

ARGUMENTS

ON OUTPUT

IERROR
an error parameter
= 0, if the test is successful,
= 1, otherwise

I/O
If the test is successful, the message
AREAS TEST SUCCESSFUL . . . SEE PLOT TO VERIFY PERFORMANCE
is written on unit 6.

PRECISION
Single.

REQUIRED LIBRARY
AREAS, SPPS

FILES

REQUIRED GKS LEVEL
OA

LANGUAGE
FORTRAN

HISTORY

ALGORITHM
TAREAS constructs and colors a very simple picture illustrating the use of all of the routines in the package.

PORTABILITY
FORTRAN 77

Define an array in which to construct the area map.

DIMENSION IAM(5000)

Define the arrays needed for edge-coordinate data.

DIMENSION XCA(73), YCA(73)

Dimension the arrays needed by ARSCAM and ARDRLN for X/Y coordinates.

DIMENSION XCS(150), YCS(150)
C Dimension the arrays needed by ARSCAM and ARDRLN for area and group identifiers.
C
DIMENSION IAI(2),IAG(2)
C Define the RGB color triples needed below.
C
DIMENSION RGB(3,15)
C Define a set of indices to order the colors.
C
DIMENSION IOC(15)
C Define an array for GKS aspect source flags.
C
DIMENSION IF(13)
C Declare the routine which will color the areas.
C
EXTERNAL COLRAM
C Declare the routine which will draw lines over the circle.
C
EXTERNAL COLRLN
C
DATA RGB / 0.70 , 0.70 , 0.70 , 
+ 0.75 , 0.50 , 1.00 , 
+ 0.50 , 0.00 , 1.00 , 
+ 0.00 , 0.00 , 1.00 , 
+ 0.00 , 0.50 , 1.00 , 
+ 0.00 , 1.00 , 1.00 , 
+ 0.00 , 1.00 , 0.60 , 
+ 0.00 , 1.00 , 0.00 , 
+ 0.70 , 1.00 , 0.00 , 
+ 1.00 , 1.00 , 0.00 , 
+ 1.00 , 0.75 , 0.00 , 
+ 1.00 , 0.38 , 0.38 , 
+ 1.00 , 0.00 , 0.38 , 
+ 1.00 , 0.00 , 0.00 , 
+ 1.00 , 1.00 , 1.00 / 
DATA IOC / 6,2,5,12,10,11,1,3,4,8,9,7,13,14,15 / 
C Declare the constant for converting from degrees to radians.
C
DATA DTR / .017453292519943 / 
C Set the aspect source flags for FILL AREA INTERIOR STYLE and for FILL AREA STYLE INDEX to "individual".
C
CALL GQASF (IE,IF)
IF(11)=1
IF(12)=1
CALL GSASF (IF)
C
C Force solid fill.
C
CALL GSFAIS (1)
C
C Define 15 different color indices.
C
CALL SETUSV ('IM', 15)
C
DO 101 J=1,15
   I=IOC(J)
   IF (RGB(1,I).GE.RGB(2,I).AND.RGB(1,I).GE.RGB(3,I)) THEN
      IND=1
      CALL SETUSV ('IR',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
      CALL SETUSV ('IG',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
      CALL SETUSV ('IB',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
      CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
   ELSE IF (RGB(2,I).GE.RGB(1,I) .AND.RGB(2,I).GE.RGB(3,I)) THEN
      IND=2
      CALL SETUSV ('IR',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
      CALL SETUSV ('IG',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
      CALL SETUSV ('IB',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
      CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
   ELSE IF (RGB(3,I).GE.RGB(1,I).AND.RGB(3,I).GE.RGB(2,I)) THEN
      IND=3
      CALL SETUSV ('IR',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
      CALL SETUSV ('IG',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
      CALL SETUSV ('IB',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
      CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
   END IF
101 CONTINUE
C
C Initialize the area map.
C
CALL ARINAM (IAM,5000)
C
C Add edges to the area map.
C
CALL SET (.01,.99,.01,.99,-1.,1.,-1.,1.,1)
C
C First, define a circle, using group 1 edges. After this step, the
area inside the circle has area identifier zero and the area outside
has area identifier -1.
C
DO 102 ING=1,73
   ANG=DTR*REAL(5*(ING-1))
   XCA(ING)=COS(ANG)
   YCA(ING)=SIN(ANG)
102 CONTINUE
CALL AREDAM (IAM,XCA,YCA,73,1,0,-1)
C
C Add lines splitting the circle into wedges. The area identifiers
for the wedges are added to the area map with this step.
C
XCA(1)=0.
YCA(1)=0.
DO 103 ING=1,15
   ANG=DTR*REAL(24*(ING-1))
   XCA(2)=COS(ANG)
   YCA(2)=SIN(ANG)
CALL AREDAM (IAM,XCA,YCA,2,1,ING,MOD(ING+13,15)+1)
103 CONTINUE
C
C Now, put in another, smaller, off-center circle, using a group 2
C edge. The interior of the circle has area identifier 1 and the
C exterior of the circle has group identifier 2.
C
DO 104 ING=1,73
   ANG=DTR*REAL(5* (ING-1))
   XCA(ING)=.25+.5*COS(ANG)
   YCA(ING)=.25+.5*SIN(ANG)
104 CONTINUE
CALL AREDAM (IAM,XCA,YCA,73,2,1,2)
C
C Pre-process the area map.
C
CALL ARPRAM (IAM,0,0,0)
C
C Compute and print the amount of space used in the area map.
C
   ISU=5000-(IAM(6)-IAM(5)-1)
   PRINT *, 'SPACE USED IN AREA MAP IS ',ISU
C
C Color the areas defined.
C
CALL ARSCAM (IAM,XCS,YCS,150,IAI,IAG,2,COLRAM)
C
C In contrasting colors, draw three stars on the plot.
C
DO 105 I=1,3
   IF (I.EQ.1) THEN
      XCN=-.5
      YCN=+.5
   ELSE IF (I.EQ.2) THEN
      XCN=-.5
      YCN=-.5
   ELSE IF (I.EQ.3) THEN
      XCN=+.5
      YCN=-.5
   END IF
   XCA(1)=XCN+.25*COS( 162.*DTR)
   YCA(1)=YCN+.25*SIN( 162.*DTR)
   XCA(2)=XCN+.25*COS( 18.*DTR)
   YCA(2)=YCN+.25*SIN( 18.*DTR)
   XCA(3)=XCN+.25*COS(-126.*DTR)
   YCA(3)=YCN+.25*SIN(-126.*DTR)
   XCA(4)=XCN+.25*COS( 90.*DTR)
   YCA(4)=YCN+.25*SIN( 90.*DTR)
   XCA(5)=XCN+.25*COS(-54.*DTR)
   YCA(5)=YCN+.25*SIN(-54.*DTR)
   XCA(6)=XCN+.25*COS( 162.*DTR)
   YCA(6)=YCN+.25*SIN( 162.*DTR)
   CALL ARDRLN (IAM,XCA,YCA,6,XCS,YCS, 150,IAI,IAG,2,COLRLN)
C Draw a spiral of points in the blanked-out circle, using the colors C from edge group 1.
C
ICF=1
DO 108 ING=1,1500
   RAD=REAL(ING)/1000.
   ANG=DTR*REAL(ING-1)
   XCD=.25+.5*RAD*COS(ANG)
   YCD=.25+.5*RAD*SIN(ANG)
   CALL ARGTAI (IAM,XCD,YCD,IAI,IAG,2,NAI,ICF)
   ITM=1
   DO 106 I=1,NAI
      IF (IAI(I).LT.O) ITM=0
   106 CONTINUE
   IF (ITM.NE.O) THEN
      IT1=0
      IT2=0
      DO 107 I=1,NAI
         IF (IAG(I).EQ.1) IT1=IAI(I)
         IF (IAG(I) .EQ.2) IT2=IAI(I)
      107 CONTINUE
      IF (IT1.GT.O.AND.IT2.EQ.1) THEN
         CALL SETUSV ('II',IT1)
         CALL POINT (XCD,YCD)
      END IF
   END IF
   ICF=0
108 CONTINUE
C
C Advance the frame.
C
CALL FRAME
C
C Done.
C
IERROR=0
WRITE (6,1001)
C
RETURN
C
1001 FORMAT (' AREAS TEST SUCCESSFUL',24X,
           'SEE PLOT TO VERIFY PERFORMANCE')
C
END
SUBROUTINE COLRAM (XCS,YCS,NCS,IAI,IAG,NAI)
DIMENSION XCS(*),YCS(*),IAI(*),IAG(*)
ITM=1
DO 101 I=1,NAI
   IF (IAI(I).LT.O) ITM=0
101 CONTINUE
   IF (ITM.NE.O) THEN
IT1=0
DO 102 I=1,NAI
  IF (IAG(I).EQ.1) IT1=IAI(I)
  IF (IAG(I).EQ.2) IT2=IAI(I)
102 CONTINUE
  IF (IT1.GT.0.AND. IT2.NE.1) THEN
    CALL SETUSV ('II', IT1)
    CALL GFA (NCS-1, XCS, YCS)
  END IF
END IF
END IF
RETURN
END

SUBROUTINE COLRLN (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
ITM=1
DO 101 I=1,NAI
  IF (IAI(I).LT.0) ITM=0
101 CONTINUE
  IF (ITM.NE.0) THEN
    IT1=0
    IT2=0
    DO 102 I=1,NAI
      IF (IAG(I).EQ.1) IT1=IAI(I)
      IF (IAG(I).EQ.2) IT2=IAI(I)
102 CONTINUE
      IF (IT1.GT.0.AND. IT2.NE.1) THEN
        CALL SETUSV ('II', MOD(IT1+3,15)+1)
        CALL GPL (NCS, XCS, YCS)
      END IF
    END IF
    RETURN
  END IF
END

Important Note: The example plot is reproduced here in black and white. If the code above were used to drive a color output device, the output would be in color. Each wedge should be a different color, the circle should be black, and the stars and swirl should be outlined in more than one color each.
AUTOGRAPH Example

SUBROUTINE TAUTOG (IERROR)
C
C PURPOSE
To provide a simple demonstration of
the AUTOGRAPH package.
C
C USAGE
CALL TAUTOG (IERROR)
C
C ARGUMENTS
C
ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
I/O
If the test is successful, the message
AUTOGRAPH TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 4
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED ROUTINES
AUTOGRAPH, DASHCHAR
C
C REQUIRED GKS LEVEL
OA
C
C HISTORY
Originally written in April, 1979.
Converted to FORTRAN 77 and GKS in Feb., 1985.
C
C ALGORITHM
TAUTOG computes data which is plotted in single
calls to each of the entries
EZY, EZXY, EZMY, AND EZMXY.
C
C On 3 of the plots, routines AGSETE, AGSETI,
and AGSETP are called to specify Y-axis labels
or to introduce log scaling.
C
REAL X(21), Y1D(21), Y2D(21,5)
C
X contains the abscissae for the plots produced by EZXY and
EZMXY, Y1D contains the ordinate values for the plots produced by
EZXY and EZY, and Y2D contains the ordinate values for the plots
produced by EZMY and EZMXY.

August 1987
Frame 1 -- EZY entry of AUTOGRAPH.

C Fill Y1D array for entry EZY.

DO 10 I=1,21
    Y1D(I) = EXP(-.1*FLOAT(I))*COS(FLOAT(I)*.5)
10 CONTINUE

Entry EZY plots Y1D as a function of a set of continuous integers.

DEMONSTRATING EZY ENTRY OF AUTOGRAPH

CALL EZY (Y1D(1),21,'DEMONSTRATING EZY ENTRY OF AUTOGRAPH$')
Frame 2 -- EZXY entry of AUTOGRAPH.

Fill X and Y1D arrays for entry EZXY.

DO 20 I=1,21
   X(I) = FLOAT(I-1)*.314
   Y1D(I) = X(I)+COS(X(I))*2.0
20 CONTINUE

Set AUTOGRAPH control parameters for Y-axis label "X+COS(X)*2"

CALL AGSETC('LABEL/NAME.', 'L')
CALL AGSETI('LINE/NUMBER.', 100)
CALL AGSETC('LINE/TEXT.', 'X+COS(X)*2$')

Entry EZXY plots contents of X-array vs. Y1D-array.

DEMONSTRATING EZXY ENTRY OF AUTOGRAPH

CALL EZXY (X,Y1D,21,'DEMONSTRATING EZXY ENTRY IN AUTOGRAPH$')
Frame 3 -- EZMY entry of AUTOGRAPH.

Fill Y2D array for entry EZMY.

DO 40 I=1,21
   T = .5*FLOAT(I-1)
   DO 30 J=1,5
      Y2D(I,J) = EXP(-.5*T)*COS(T)/FLOAT(J)
   30 CONTINUE
40 CONTINUE

Set the AUTOGRAPH control parameters for Y-axis label

CALL AGSETC('LABEL/NAME.', 'L')
CALL AGSETI ('LINE/NUMBER.', 100)
CALL AGSETC('LINE/TEXT.', 'EXP(-X/2)*COS(X)*SCALE$')

Use the AUTOGRAPH control parameter for integers to specify the alphabetic set of dashed line patterns.

CALL AGSETI ('DASH/SELECTOR.', -1)

Use the AUTOGRAPH control parameter for integers to specify the graph drawn is to be logarithmic in the X-axis.

CALL AGSETI ('X/LOGARITHMIC.', 1)

Entry EZMY plots multiple arrays as a function of continuous integers.

DEMONSTRATING EZMY ENTRY OF AUTOGRAPH

CALL EZMY (Y2D, 21, 5, 10, 'DEMONSTRATING EZMY ENTRY OF AUTOGRAPH$')
DEMONSTRATING EZMY ENTRY OF AUTOGRAPH

\[ \text{Figure: } \exp(-x/2) \times \cos(x) \times \text{SCALE} \]

\[ x \in [10^0, 10^1] \]

August 1987
C
C Frame 4 -- EZMXY entry of AUTOGRAPH.
C
C Fill Y2D array for EZMXY.
C
DO 60 I=1,21
   DO 50 J=1,5
      Y2D(I,J) = X(I)**J+COS(X(I))
      CONTINUE
50 CONTINUE
60 CONTINUE
C
C Set the AUTOGRAPH control parameters for Y-axis label
C
   CALL AGSETC('LABEL/NAME.', 'L')
   CALL AGSETI('LINE/NUMBER.', 100)
   CALL AGSETC('LINE/TEXT.', 'X**J+COS(X)\$')
C
C Use the AUTOGRAPH control parameter for integers to specify the
C alphabetic set of dashed line patterns.
C
   CALL AGSETI('DASH/SELECTOR.', -1)
C
C Use the AUTOGRAPH control parameter for integers to specify the
C graph have a linear X-axis and a logarithmic Y-axis.
C
   CALL AGSETI('X/LOGARITHMIC.', 0)
   CALL AGSETI('Y/LOGARITHMIC.', 1)
C
C Entry EZMXY plots multiple Y arrays as a function of a single
C or multiple X arrays.
C
   DEMONSTRATING EZMXY ENTRY OF AUTOGRAPH
C
   CALL EZMXY (X,Y2D,21,5,21,
      'DEMONSTRATING EZMXY ENTRY OF AUTOGRAPH\$')
C
C Note that AUTOGRAPH makes its own FRAME advance calls.
C
   IERROR = 0
   WRITE (6,1001)
C
RETURN
C
1001 FORMAT (' AUTOGRAPH TEST EXECUTED--SEE PLOTS TO CERTIFY')
C
END
DEMONSTRATING EZMXY ENTRY OF AUTOGRAPH
Standard Contouring

CONREC + DASHCHAR Example

SUBROUTINE TCONRE (IERROR)

C
C PURPOSE
To provide a simple demonstration of standard
contouring of regularly spaced data using
the CONREC package.
C
C USAGE
CALL TCONRE (IERROR)
C
C ARGUMENTS
C
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
I/O
If the test is successful, the message
CONREC TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 2
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
PRECISION
Single
C
LANGUAGE
FORTRAN 77
C
REQUIRED ROUTINES
CONREC, DASHCHAR
C
REQUIRED GKS LEVEL
OA
C
ALGORITHM
The function

\[ Z(X,Y) = X + Y + \frac{1}{((X-.1)^2+Y^2+.09)} - \frac{1}{((X+.1)^2+Y^2+.09)} \]

for \( X = -1. \) to \( +1. \) in increments of \( .1 \), and
\( Y = -1.2 \) to \( +1.2 \) in increments of \( .1 \),
is computed. Then, entries EZCNTR and CONREC
are called to generate contour plots of \( Z \).
C
This is the standard version of the CONREC
family of utilities.
C
Z contains the values to be plotted.

August 1987 421
C
REAL Z(21,25)
C
C Define the position of the plot title.
C
DATA TX/.3955/, TY/.9765/
C
C Initialize the error parameter.
C
IERROR = 0
C
C Fill the 2-D array to be plotted.
C
DO 20 I=1,21
   X = .1*FLOAT(I-11)
   DO 10 J=1,25
      Y = .1*FLOAT(J-13)
      Z(I,J) = X+Y+1./((X-.10)**2+Y**2+.09) -
      1./(X+.10)**2+Y**2+.09)
   10 CONTINUE
20 CONTINUE
C
C Select normalization transformation number 0.
C
CALL GSELNT ( 0 )
C
C
C Frame 1 -- EZCNTR entry of CONREC.
C
C Entry EZCNTR requires only the array name and dimensions.
C
CALL WTSTR ( TX, TY,
   1 'DEMONSTRATION PLOT FOR EZCNTR ENTRY OF CONREC',2,0,0 )
CALL EZCNTR (Z,21,25)
Frame 2 -- CONREC entry of CONREC.

Entry CONREC allows user definition of various plot parameters.

In this example, the lowest contour level (-4.5), the highest contour level (4.5), and the increment between contour levels (0.3) are set.

```
CALL WTSTR ( TX ,TY, 
1 'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONREC',2,0,0 )
CALL CONREC (Z,21,21,25,-4.5,4.5,.3,0,0,0)
CALL FRAME

WRITE (6,1001)
RETURN

1001 FORMAT (' CONREC TEST EXECUTED--SEE PLOTS TO CERTIFY')
```

END
Standard Contouring

DEMONSTRATION PLOT FOR CONREC ENTRY OF CONREC

CONTOUR FROM -4.5000 TO 4.5000 CONTOUR INTERVAL OF 0.3000 PT3.31 = -1.9066

August 1987
CONRAN + DASHCHAR Example

SUBROUTINE TCONAN (IERROR)

C
C PURPOSE
C
To provide a simple demonstration of
standard contouring of irregularly spaced
data using the CONRAN package.
C
C USAGE
C
CALL TCONAN (IERROR)
C
C ARGUMENTS
C
IERROR

An integer variable
= 0, if the test was successful,
> 0, the test was not successful,
and the error number corresponds
to the number in the CONRAN listing.
C
C I/O
C
If the test is successful, the message
CONRAN TEST EXECUTED--SEE PLOTS TO CERTIFY
C
C
is printed on unit 6. In addition, 2
frames are produced on the machine graphics
device. The first plot is the contour plot.
The second plot shows the triangulation of the
data. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
C
Single
C
C LANGUAGE
C
FORTRAN 77
C
C REQUIRED ROUTINES
C
CONRAN, CONTERP, CONCOM, DASHCHAR
C
C REQUIRED GKS LEVEL
C
OA
C
C ALGORITHM
C
A sparse dataset is defined in DATA statements.
Options are selected to produce a plot title
and display the triangulation generated by the
interpolation routines. Default options
include a message at the bottom of the plot
and a plot perimeter.
C
C
This is the standard version of the CONRAN
family of utilities.
C
C
The version of the package which produces
smoothed contours is created by simply
loading this package with DASHSMTH rather

than DASHCHAR.

C

COMMON /RANINT/ IRANMJ, IRANMN, IRANTX

C Set up the scratch arrays needed by CONRAN.

C

DIMENSION WK(221), IWK(744), SCR(1600)

C Dimension arrays to hold the sparse dataset.

C

DIMENSION XD(17), YD(17), ZD(17)

C

C Define the dataset.

C

DATA XD(1), XD(2), XD(3), XD(4), XD(5), XD(6), XD(7), XD(8),
1 XD(9), XD(10), XD(11), XD(12), XD(13), XD(14), XD(15),
2 XD(16), XD(17)
3 /3., 10., 18., 8., 10., 5., 1., 15., 20.,
4 5., 15., 10., 7., 13., 16./

C

DATA YD(1), YD(2), YD(3), YD(4), YD(5), YD(6), YD(7), YD(8),
1 YD(9), YD(10), YD(11), YD(12), YD(13), YD(14), YD(15),
2 YD(16), YD(17)
3 /3., 18., 18., 3., 18., 10., 1., 5., 10., 5., 10.,
4 15., 15., 15., 20., 20., 8./

C

DATA ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6), ZD(7), ZD(8),
1 ZD(9), ZD(10), ZD(11), ZD(12), ZD(13), ZD(14), ZD(15),
2 ZD(16), ZD(17)
4 1., 1., 1., 1., 1., 25./

C

C Define the number of points in the dataset.

C

DATA NDP/17/

C

C Set the PORT error handling routine to the recover mode.

C

CALL ENTSR(IROLD, 1)

C

C Define the plot title.

C

CALL CONOP4('TLE=ON', 'DEMONSTRATION PLOT FOR CONRAN', 29, 0)

C

C Test for an error condition.

C

IF (NERRO(IERROR).NE.0) GO TO 100

C

C No error encountered.

C

C

C Set the option to generate the triangulation display.

C
CALL CONOP1('TRI=ON')
C
C Again, test for an error condition.
C
IF (NERRO(IERROR).NE.0) GO TO 100
C
C No error encountered.
C
C Call CONRAN to contour the data.
C
CALL CONRAN(XD,YD,ZD,NDP,WK,IWK,SCR)
C
C Again, test for an error condition.
C
IF (NERRO(IERROR).NE.0) GO TO 100
C
C No error encountered.
C
C Advance the frame. CONRAN does not do so internally.
C
CALL FRAME
C
C Print the successful completion message.
C
WRITE(6,10)
10 FORMAT(' CONRAN TEST EXECUTED--SEE PLOTS TO CERTIFY')
RETURN
C
C If an error was encountered call the PORT error print routine.
C This is only necessary if you are in recover mode, else the message
C is printed automatically.
C
100 CALL EPRIN
RETURN
END
If option TRI of CONREC is set to "ON" as in the above example using:

C
C Set the option to generate the triangulation display.
C
CALL CONOP1('TRI=ON')

Then, the following plot is also produced:

August 1987
Quick Contouring

CONRECQCK + DASHLINE Example

SUBROUTINE TCNQCK (IERROR)
C
C PURPOSE
To provide a simple demonstration of
the CONRECQCK package.
C
C USAGE
CALL TCNQCK (IERROR)
C
C ARGUMENTS
C
C I/O
If the test is successful, the message
C
CONRECQCK TEST EXECUTED--SEE PLOTS TO CERTIFY
C
is printed on unit 6. In addition, 2
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED ROUTINES
CONRECQCK, DASHLINE
C
C REQUIRED GKS LEVEL
OA
C
C ALGORITHM
The function
C
\[ Z(X,Y) = X + Y + 1/((X-.1)**2+Y**2+.09) \]
\[ -1/((X+.1)**2+Y**2+.09) \]
C
for \( X = -1 \) to +1. in increments of .1, and
Y = -1.2 to +1.2 in increments of .1,
is computed. Then, entries EZCNTR and CONREC
are called to generate contour plots of \( Z \).
C
This is a quick version of the CONREC utility.
C
Z contains the values to be plotted.
C
REAL Z(21,25)
C Define the position of the plot title.
C
DATA TX/.4267/, TY/.9765/
C
C Initialize the error parameter.
C
IERROR = 0
C
C Fill a 2-D array to be plotted.
C
DO 20 I=1,21
   X = .1*FLOAT(I-11)
   DO 10 J=1,25
      Y = .1*FLOAT(J-13)
      Z(I,J) = X+Y+1./((X-.10)**2+Y**2+.09) -
               1./((X+.10)**2+Y**2+.09)
   10 CONTINUE
20 CONTINUE
C
C Select normalization transformation number 0.
C
CALL GSELNT (0)
C
C Frame 1 -- EZCNTR entry of CONRECQCK.
C
C Entry EZCNTR requires only the array name and dimensions.
C
CALL WTSTR ( TX, TY,  
       'DEMONSTRATION PLOT FOR EZCNTR ENTRY OF CONRECQCK',2,0,0 )
CALL EZCNTR (Z,21,25)
Quick Contouring

DEMONSTRATION PLOT FOR EZCNTR ENTRY OF CONRECQCK

CONTOUR FROM -4.5000 TO 4.5000 CONTOUR INTERVAL OF 0.50000 PT13,3) = -1.9066

August 1987
Frame 2 -- CONREC entry of CONRECQCK.

Entry CONREC allows user definition of various plot parameters.

In this example, the lowest contour level (-4.5), the highest contour level (4.5), and the increment between contour levels (0.3) are set.

CALL WTSTR (TX, TY,
1    'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECQCK',
2    2,0,0)
CALL CONREC (Z, 21, 21, 25, -4.5, 4.5, .3, 0, 0, 0)
CALL FRAME

WRITE (6,1001)
RETURN

1001 FORMAT (' CONRECQCK TEST EXECUTED--SEE PLOTS TO CERTIFY')
END
CONRAQ + DASHLINE Example

SUBROUTINE TCONAQ (IERROR)

C
C PURPOSE
To provide a simple demonstration of
the CONRAQ package.
C
C USAGE
CALL TCONAQ (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
> 0, the test was not successful,
and the error number corresponds
to the number in the CONRAQ listing.
C
C I/O
If the test is successful, the message
CONRAQ TEST EXECUTED--SEE PLOTS TO CERTIFY
C
C is printed on unit 6. In addition, 2 output
frames are produced on the machine graphics
device. The first plot is the contour plot.
The second plot shows the triangulation of the
data. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED Routines
CONRAQ, CONTERP, DASHLINE
C
C REQUIRED GKS LEVEL
OA
C
C ALGORITHM
A sparse dataset is defined in DATA statements.
Options are selected to produce a plot title
and display the triangulation generated by the
interpolation routines. Default options
include a message at the bottom of the plot
and a plot perimeter.
C
This is the quick version of the CONRAN
family of utilities.
C

COMMON /RAQINT/ IRAQMJ, IRAQMN, IRAQTX
C
Set up the scratch arrays needed by CONRAQ.
C
August 1987
C Dimension arrays to hold the sparse dataset.
C
DIMENSION XD(17), YD(17), ZD(17)
C
C Define the dataset.
C
DATA XD(1), XD(2), XD(3), XD(4), XD(5), XD(6), XD(7), XD(8),
1 XD(9), XD(10), XD(11), XD(12), XD(13), XD(14), XD(15),
2 XD(16), XD(17)
3 /3., 3., 10., 18., 18., 10., 10., 5., 1., 15., 20.,
4 5., 15., 10., 7., 13., 16./
C
DATA YD(1), YD(2), YD(3), YD(4), YD(5), YD(6), YD(7), YD(8),
1 YD(9), YD(10), YD(11), YD(12), YD(13), YD(14), YD(15),
2 YD(16), YD(17)
3 /3., 18., 18., 3., 18., 10., 1., 5., 10., 5., 10.,
4 15., 15., 15., 20., 20., 8./
C
DATA ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6), ZD(7), ZD(8),
1 ZD(9), ZD(10), ZD(11), ZD(12), ZD(13), ZD(14), ZD(15),
2 ZD(16), ZD(17)
4 1., 1., 1., 1., 1., 25./
C
C Define the number of points in the dataset.
C
DATA NDP/17/
C
C Set the PORT error handling routine to the recover mode.
C
CALL ENTSR (IROLD, 1)
C
C Define the plot title.
C
CALL CONOP4 ('TLE=ON', 'DEMONSTRATION PLOT FOR CONRAQ', 29, 0)
C
C Test for an error condition.
C
IF (NERRO (IERROR) .NE. 0) GO TO 100
C
C No error encountered.
C
C Set the option to generate the triangulation display.
C
CALL CONOP1 ('TRI=ON')
C
C Test for an error condition.
C
IF (NERRO (IERROR) .NE. 0) GO TO 100
C
C No error encountered.
Call CONRAQ to contour the data.

CALL CONRAQ(XD, YD, ZD, NDP, WK, IWK)

Test for an error condition.

IF (NERRO(IERROR).NE.0) GO TO 100

No error encountered.

Advance the frame. CONRAQ does not do this internally.

CALL FRAME

Print the all is well message.

WRITE(6,10)

10 FORMAT(' CONRAQ TEST EXECUTED--SEE PLOTS TO CERTIFY')

RETURN

If an error was encountered call the PORT error print routine. This is only necessary if you are in recover mode, else the message is printed automatically.

CALL EPRIN

RETURN

END
If option TRI of CONREC is set to "ON" as in the above example using:
C
CALL CONOP1('TRI=ON')
Then, the following plot is also produced:
Smooth Contouring

CONRECSMTH + DASHSMTH Example
(CONREC + DASHSMTH)

Important Note: The CONRECSMTH package consists of the CONREC package together with the DASHSMTH package. There is no separate file on the distribution tape called CONRECSMTH, since it can be obtained from CONREC and DASHSMTH. The CONREC package uses DASHCHAR by default.

SUBROUTINE TCNSMT (IERROR)

C
C PURPOSE
C To provide a simple demonstration of
C the CONRECSMTH package.
C
C USAGE
C CALL TCNSMT (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
C IERROR
C An integer variable
C = 0, if the test was successful,
C = 1, otherwise
C
C I/O
C If the test is successful, the message
C CONREC SMOOTH TEST EXECUTED--SEE PLOTS TO CERTIFY
C is printed on unit 6. In addition, 2
C frames are produced on the machine graphics
C device. In order to determine if the test
C was successful, it is necessary to examine
C the plots.
C
C PRECISION
C Single
C
C LANGUAGE
C FORTRAN 77
C
C REQUIRED ROUTINES
C CONREC, DASHSMTH
C
C REQUIRED GKS LEVEL
C OA
C
C ALGORITHM
C The function
C
C Z(X,Y) = X + Y + 1./(X-.1)**2+Y**2+.09)
C -1./(X+.1)**2+Y**2+.09)
C
C for X = -1. to +1. in increments of .1, and
C Y = -1.2 to +1.2 in increments of .1,
C is computed. Then, entries EZCNTR and CONREC
C are called to generate contour plots of Z.

August 1987
C This is a smoothed curve version of the CONREC family of utilities.

C

C Z contains the values to be plotted.
C

REAL Z(21,25)

C Define the center of a plot title string on a square grid of size 0. to 1.
C

DATA TX/.4267/, TY/.9765/

C Initialize the error parameter.
C

IERROR = 0

C Fill a 2-D array to be plotted.
C

DO 20 I=1,21
   X = .1*FLOAT(I-11)
   DO 10 J=1,25
      Y = .1*FLOAT(J-13)
      Z(I,J) = X+Y+1./((X-.10)**2+Y**2+.09) -
                    1./((X+.10)**2+Y**2+.09)
   10 CONTINUE
20 CONTINUE

C Select normalization transformation number 0.
C

CALL GSELNT (0)

C Frame 1 -- EZCNTR entry of CONRECSMTH.
C

C Entry EZCNTR requires only the array name and dimensions.
C

CALL WTSTR ( TX, TY, 1 'DEMONSTRATION PLOT FOR EZCNTR ENTRY OF CONRECSMTH',2,0,0 )
CALL EZCNTR (Z,21,25)
Smooth Contouring

DEMONSTRATION PLOT FOR EZCNTR ENTRY OF CONRECSMTH

CONTOUR FROM -4.5000 TO 4.5000 CONTOUR INTERVAL OF 0.50000 PT(3,3) = -1.9066

August 1987
Frame 2 -- CONREC entry of CONRECSMTH.

Entry CONREC allows user definition of various plot parameters.

In this example, the lowest contour level (-4.5), the highest contour level (4.5), and the increment between contour levels (0.3) are set.

CALL WTSTR (TX, TY,
1 'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSMTH',
2 2, 0, 0)
CALL CONREC (Z, 21, 21, 25, -4.5, 4.5, .3, 0, 0, 0)
CALL FRAME

WRITE (6, 1001)
RETURN

1001 FORMAT (' CONREC SMOOTH TEST EXECUTED--SEE PLOTS TO CERTIFY')
END
CONRAN + DASHSMTH Example

SUBROUTINE TCONAN (IERROR)

C PURPOSE
To provide a simple demonstration of
standard contouring of irregularly spaced
data using the CONRAN package with DASHSMTH,
producing smoothed contour lines.

C USAGE
CALL TCONAN (IERROR)

C ARGUMENTS
C ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
> 0, the test was not successful,
and the error number corresponds
to the number in the CONRAN listing.

C I/O
If the test is successful, the message
CONRAN TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 2
frames are produced on the machine graphics
device. The first plot is the contour plot.
The second plot shows the triangulation of the
data. In order to determine if the test
was successful, it is necessary to examine
the plots.

C PRECISION
Single

C LANGUAGE
FORTRAN 77

C REQUIRED ROUTINES
CONRAN, CONTERP, CONCOM, DASHSMTH

C REQUIRED GKS LEVEL
OA

C ALGORITHM
A sparse dataset is defined in DATA statements.
Options are selected to produce a plot title
and display the triangulation generated by the
interpolation routines. Default options
include a message at the bottom of the plot
and a plot perimeter.

This is the standard version of the CONRAN
family of utilities.

The version of the package which produces
smoothed contours is created by simply

August 1987
COMMON /RANINT/ IRANMJ, IRANMN, IRANTX

C Set up the scratch arrays needed by CONRAN.
C
DIMENSION WK(221), IWK(744), SCR(1600)

C Dimension arrays to hold the sparse dataset.
C
DIMENSION XD(17), YD(17), ZD(17)

C Define the dataset.

DATA XD(1), XD(2), XD(3), XD(4), XD(5), XD(6), XD(7), XD(8),
    XD(9), XD(10), XD(11), XD(12), XD(13), XD(14), XD(15),
    XD(16), XD(17)

1 /3., 3., 10., 18., 18., 10., 10., 5., 1., 15., 20.,
2   5., 15., 10., 7., 13., 16./

C DATA YD(1), YD(2), YD(3), YD(4), YD(5), YD(6), YD(7), YD(8),
    YD(9), YD(10), YD(11), YD(12), YD(13), YD(14), YD(15),
    YD(16), YD(17)

1 /3., 18., 18., 3., 18., 10., 1., 5., 10., 5., 10.,
2   15., 15., 15., 20., 20., 8./

C DATA ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6), ZD(7), ZD(8),
    ZD(9), ZD(10), ZD(11), ZD(12), ZD(13), ZD(14), ZD(15),
    ZD(16), ZD(17)

2   1., 1., 1., 1., 1., 25./

C Define the number of points in the dataset.
C
DATA NDP/17/

C Set the PORT error handling routine to the recover mode.
C
CALL ENTSR(IROLD, 1)

C Define the plot title.
C
CALL CONOP4('TLE=ON','DEMONSTRATION PLOT FOR CONRAN', 29, 0)

C Test for an error condition.
C
IF (NERRO(IERROR).NE.O) GO TO 100

C No error encountered.
C
C Set the option to generate the triangulation display.
CALL CONOP1('TRI=ON')

C Again, test for an error condition.
C
IF (NERRO(IERROR).NE.O) GO TO 100

C No error encountered.
C
Call CONRAN to contour the data.
C
CALL CONRAN(XD,YD,ZD,NDP,WK,IWK,SCR)

C Again, test for an error condition.
C
IF (NERRO(IERROR).NE.O) GO TO 100

C No error encountered.
C
C Advance the frame. CONRAN does not do so internally.
C
CALL FRAME

C Print the successful completion message.
C
WRITE(6,10)
10 FORMAT(' CONRAN TEST EXECUTED--SEE PLOTS TO CERTIFY')

C If an error was encountered call the PORT error print routine.
C This is only necessary if you are in recover mode, else the message
C is printed automatically.
C
100 CALL EPRIN
RETURN
END
If option TRI of CONREC is set to "ON" as in the above example using:

C
C Set the option to generate the triangulation display.
C
CALL CONOP1('TRI=ON')

Then, the following plot is also produced:
Super Contouring

CONRECSPR + DASHSUPR Example

SUBROUTINE TCNSUP (IERROR)
C
C PURPOSE
C
To provide a simple demonstration of
the CONRECSPR package.
C
C USAGE
C
CALL TCNSUP (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
C
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
C I/O
C
If the test is successful, the message
CONRECSPR TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 2 output
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
C
Single
C
C LANGUAGE
C
FORTRAN 77
C
C REQUIRED ROUTINES
C
CONRECSPR, DASHSUPR
C
C REQUIRED GKS LEVEL
C
OA
C
C ALGORITHM
C
The function
Z(X,Y) = X + Y + 1./((X-.1)**2+Y**2+.09)
-1./((X+.1)**2+Y**2+.09)
C
for X = -1. to +1. in increments of .1, and
Y = -1.2 to +1.2 in increments of .1,
is computed. Then, entries EZCNTR and CONREC
are called to generate contour plots of Z.
C
C
This is the highest quality entry of the
CONREC family of utilities.
C
C Z contains the values to be plotted.
C
REAL Z(21,25)

C Define the position of the title.
C DATA TX/.4267/, TY/.9765/
C
C Initialize the error parameter.
C IERROR = 0
C
C Fill a 2-D array to be plotted.
C DO 20 I=1,21
   X = .1*FLOAT(I-11)
   DO 10 J=1,25
      Y = .1*FLOAT(J-13)
      Z(I,J) = X+Y+1./((X-.10)**2+Y**2+.09)-
         1./((X+.10)**2+Y**2+.09)
   10 CONTINUE
20 CONTINUE

C Select normalization transformation number 0.
C CALL GSELNT (0)
C
C Frame 1 -- EZCNTR entry of CONRECSPR.
C Entry EZCNTR requires only the array name and dimensions.
C CALL WTSTR ( TX, TY,
   1 'DEMONSTRATION PLOT FOR EZCNTR ENTRY OF CONRECSPR',2,0,0 )
CALL EZCNTR (Z,21,25)
DEMOnSTRATION PLOT FOR EZCNTR ENTRY OF CONRECSPR

CONTOUR FROM -4.5000 TO 4.5000 CONTOUR INTERVAL OF 0.50000

PT(10.31x = -1.9066

August 1987
Frame 2 -- CONREC entry of CONRECSPR.

Entry CONREC allows user definition of various plot parameters.

In this example, the lowest contour level (-4.5), the highest contour level (4.5), and the increment between contour levels (0.3) are set.

```fortran
CALL WTSTR (TX,TY,
1     'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSPR',
2     2,0,0)
CALL CONREC (Z,21,21,25,-4.5,4.5, .3,0,-1, 0)
CALL FRAME

WRITE (6,1001)
RETURN

1001 FORMAT (' CONRECSPR TEST EXECUTED--SEE PLOTS TO CERTIFY')
END
```

DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSPR

CONTOUR FROM -4.5000 TO 4.5000 CONTOUR INTERVAL OF 0.30000 PT(3,31- -1.9066
CONRAS + DASHSUPR Example

SUBROUTINE TCONAS (IERROR)
C
C PURPOSE
C To provide a simple demonstration of
C CONRAS, the super entry of the CONRAN
C package.
C
C USAGE
C CALL TCONAS (IERROR)
C
C ARGUMENTS
C IERROR
C An integer variable
C = 0, if the test was successful,
C > 0, the test was not successful,
C and the error number corresponds
C to the number in the CONRAS listing.
C
C ON OUTPUT
C If the test is successful, the message
C
C CONRAS TEST EXECUTED--SEE PLOTS TO CERTIFY
C
C is printed on unit 6. In addition, 2
C frames are produced on the machine graphics
C device. The first plot is the contour plot.
C The second plot shows the triangulation of the
C data. In order to determine if the test
C was successful, it is necessary to examine
C the plots.
C
C PRECISION
C Single
C
C LANGUAGE
C FORTRAN 77
C
C REQUIRED ROUTINES
C CONRAS, CONTERP, CONCOM, DASHSUPR
C
C REQUIRED GKS LEVEL
C OA
C
C ALGORITHM
C A sparse dataset is defined in DATA statements.
C Options are selected to produce a plot title
C and display the triangulation generated by the
C interpolation routines. Default options
C include a message at the bottom of the plot
C and a plot perimeter.
C
C This is the super version of the CONRAN
C family of utilities. It is created by
C loading this package with DASHSUPR rather
C than DASHCHAR.
C
COMMON /RANINT/ IRANMJ, IRANMN, IRANTX

August 1987
C Set up the scratch arrays needed by CONRAS.
C
DIMENSION WK(221), IWK(744), SCR(1600)
C
Dimension arrays to hold the sparse dataset.
C
DIMENSION XD(17), YD(17), ZD(17)
C
Define the dataset.
C
DATA XD(1), XD(2), XD(3), XD(4), XD(5), XD(6), XD(7), XD(8),
1 XD(9), XD(10), XD(11), XD(12), XD(13), XD(14), XD(15),
2 XD(16), XD(17)
3 /3., 3., 10., 18., 18., 10., 10., 5., 1., 15., 20.,
4 5., 15., 10., 7., 13., 16./
C
DATA YD(1), YD(2), YD(3), YD(4), YD(5), YD(6), YD(7), YD(8),
1 YD(9), YD(10), YD(11), YD(12), YD(13), YD(14), YD(15),
2 YD(16), YD(17)
3 /3., 18., 18., 3., 18., 10., 1., 5., 10., 5., 10.,
4 15., 15., 15., 20., 20., 8./
C
DATA ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6), ZD(7), ZD(8),
1 ZD(9), ZD(10), ZD(11), ZD(12), ZD(13), ZD(14), ZD(15),
2 ZD(16), ZD(17)
4 1., 1., 1., 1., 1., 25./
C
Define the number of points in the dataset.
C
DATA NDP/17/
C
Set the PORT error handling routine to the recover mode.
C
CALL ENTSR(IROLD, 1)
C
Define the plot title.
C
CALL CONOP4('TLE=ON', 'DEMONSTRATION PLOT FOR CONRAS', 29, 0)
C
Test for an error condition.
C
IF (NERRO(IERROR).NE.0) GO TO 100
C
No error encountered.
C
Set the option to generate the triangulation display.
C
CALL CONOP1('TRI=ON')
C
Again, test for an error condition.
IF (NERRO(IERROR).NE.0) GO TO 100

C No error encountered.
C
C Call CONRAS to contour the data.
C
CALL CONRAS(XD,YD,ZD,NDP,WK,IWK,SCR)
C
Again, test for an error condition.
C
IF (NERRO(IERROR).NE.0) GO TO 100
C
C No error encountered.
C
C Advance the frame. CONRAS does not do so internally.
C
CALL FRAME
C
C Print the successful completion message.
C
WRITE(6,10)
10 FORMAT(' CONRAS TEST EXECUTED--SEE PLOTS TO CERTIFY')
RETURN
C
C If an error was encountered call the PORT error print routine.
C This is only necessary if you are in recover mode, else the message
C is printed automatically.
C
100 CALL EPRIN
RETURN
END
If option TRI of CONREC is set to "ON" as in the above example using:

```
C
C Set the option to generate the triangulation display.
C
CALL CONOP1('TRI=ON')
```

Then, the following plot is also produced:
DASHCHAR Example

SUBROUTINE TDASHC (IERROR)
C
C PURPOSE
To provide a simple demonstration of DASHCHAR
C
C USAGE
CALL TDASHC (IERROR)
C
C ARGUMENTS
C
C IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
C I/O
If the test is successful, the message
DASHCHAR TEST EXECUTED--SEE PLOT TO CERTIFY
is printed on unit 6. In addition, 1
frame is produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plot.
C
C PRECISION
Single
C
C REQUIRED ROUTINES
DASHCHAR
C
C REQUIRED GKS LEVEL
OA
C
C LANGUAGE
FORTRAN 77
C
C ALGORITHM
TDASHC utilizes the software DASHCHAR
routines DASHDB, DASHDC, FRSTD, VECTD,
LINED, and CURVED to draw 5 curves on 1
picture using 5 different DASHCHAR patterns.
Each curve is centered about solid axis
lines and labeled with the character
representation of the DASHCHAR pattern used.
C
X contains the abscissae and Y the ordinates of the curve to be plotted.
C
DIMENSION X(31), Y(31)
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C Set a solid dash pattern, 1111111111111111 (BINARY).
C Boolean operations (using locally-implemented support
C routines) are used to support porting to hosts with 16

August 1987
455
C bit integers.
C
ISOLID = IOR (ISHIFT (32767,1), 1)
C
DO 130 K=1,5
   CALL DASHDB (ISOLID)
   ORG = 1.07 - 0.195*K
C
C Draw the central axis for each curve.
C
   CALL FRSTD (.50, ORG-0.03)
   CALL VECTD (.50, ORG+0.03)
   CALL LINED (.109, ORG, .891, ORG)
C
C Call DASHDC with a different dashed line and character combination
C for each of 5 different curves.
C
   GO TO ( 10, 20, 30, 40, 50), K
10   CALL DASHDC ('$''$''$''$''$''$''$K = 1', 10, 12)
   GO TO 60
20   CALL DASHDC ('$$$$$$''$''$$$$$$K = 2', 10, 12)
   GO TO 60
30   CALL DASHDC ('$$$$''$$$$''$$$$''K = 3', 10, 12)
   GO TO 60
40   CALL DASHDC ('$$$$$''''''''''$$$$$K = 4', 10, 12)
   GO TO 60
50   CALL DASHDC ('$$$''$$$''$$$''$$$K = 5', 10, 12)
60   CONTINUE
C
C Compute some curve coordinates and draw the curves.
C
   DO 70 I=1,31
      THETA = FLOAT(I-1)*3.1415926535897932/15.
      X(I) = 0.5+.4*COS(THETA)
      Y(I) = ORG+.075*SIN(FLOAT(K)*THETA)
70   CONTINUE
   CALL CURVED (X,Y,31)
C
C Label the curves with the character representation of the appropriate
C DASHCHAR pattern. In the pattern labels, A and D should be interpreted
C as the apostrophe and dollar sign.
C
C Locate the string at the left end, but vertically centered.
C
   CALL GSTXAL(1,3)
C
C Set the character height.
C
   CALL GSCHH(.012)
C
   ORY = ORG+.089
   GO TO ( 80, 90, 100, 110, 120), K
80   CALL GTX(.1, ORY, 'IPAT=DADADADADADADK=1')
GO TO 130
90 CALL GTX(1,ORY,'IPAT=DDDDDDADADDDDDDK=2')
GO TO 130
100 CALL GTX(1,ORY,'IPAT=DDDDADDDADDDDDAK=3')
GO TO 130
110 CALL GTX(1,ORY,'IPAT=DDDDDAAAAADDDDDK=4')
GO TO 130
120 CALL GTX(1,ORY,'IPAT=DDDADDDADDDADDDK=5')
C
130 CONTINUE
C
CALL GSTXAL(2,3)
CALL GTX (.5,.985,'DEMONSTRATION PLOT FOR DASHCHAR')
CALL GTX (.5,.015,'IN IPAT STRINGS, A AND D SHOULD BE INTERPRETED
1AS APOSTROPE AND DOLLAR SIGN')
CALL FRAME
C
IERRO = 0
WRITE (6,1001)
C
RETURN
1001 FORMAT (' DASHCHAR TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR DASHCHAR
IPAT=DADADADADADADADK=1
\[ \ldots \]
IPAT=DDDDDDADADDDDDDK=2
\[ \ldots \]
IPAT=DDDDADDDADDDDDAK=3
\[ \ldots \]
IPAT=DDDDAAAAADDDDDDK=4
\[ \ldots \]
IPAT=DDDDADDDADDDDDDK=5
IPAT STRINGS, A AND D SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SI
DASHLINE Example

SUBROUTINE TDASHL (IERROR)
C
C PURPOSE
To provide a simple demonstration of DASHLINE
C
C USAGE
CALL TDASHL (IERROR)
C
C ARGUMENTS
C
C
C
C
C
ON OUTPUT
C IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
C I/O
If the test is successful, the message
DASHLINE TEST EXECUTED--SEE PLOT TO CERTIFY
C
C is printed on unit 6. In addition, 1
C frame is produced on the machine graphics
C device. In order to determine if the test
C was successful, it is necessary to examine
C the plot.
C
C PRECISION Single
C
C REQUIRED ROUTINES DASHLINE
C
C REQUIRED GKS LEVEL OA
C
C LANGUAGE FORTRAN 77
C
C ALGORITHM TDASHL utilizes the software DASHLINE
C routines DASHDB, DASHDC, FRSTD, VECTD,
C LINED, and CURVED to draw 5 curves on 1
C picture using 5 different DASHLINE patterns.
C Each curve is centered about solid axis
C lines and labeled with the binary
C representation of the DASHLINE pattern used.
C
C X contains the abscissae and Y the ordinates of the curve to be plotted.
C
DIMENSION X(31) ,Y(31) ,IPAT(5)
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C Set a solid dash pattern, 1111111111111111 (BINARY).
C Boolean operations (using locally-implemented support
C routines) are used to support porting to hosts with 16
C bit integers.
C
    ISOLID = IOR (ISHIFT (32767,1), 1)
C
C Array IPAT contains 5 different 16-BIT dash patterns. The patterns
C are constructed with boolean operations as shown above.
C The binary representations of the patterns are
C
    0001110001111111
    1111000011110000
    1111100111111100
    1111110011110000
    1111111111111100
C
    IPAT(1) = IOR (ISHIFT (3647,1), 1)
    IPAT(2) = ISHIFT (30840,1)
    IPAT(3) = ISHIFT (32382,1)
    IPAT(4) = ISHIFT (32640,1)
    IPAT(5) = ISHIFT (32766,1)
C
    DO 70 K=1,5
        CALL DASHDB (ISOLID)
        ORG = 1.07 - 0.195*K
C
    Draw the central axis for each curve.
C
        CALL FRSTD (.50, ORG-0.03)
        CALL VECTD (.50, ORG+0.03)
        CALL LINED (.109, ORG, .891, ORG)
        CALL DASHDB (IPAT(K))
C
    Compute the curve coordinates and draw the curve.
C
        DO 10 I=1,31
            THETA = FLOAT(I-1)*3.1415926535897932/15.
            X(I) = 0.5+.4*COS(THETA)
            Y(I) = ORG+.075*SIN(FLOAT(K)*THETA)
        10 CONTINUE
        CALL CURVED (X,Y,31)
C
    Label each curve with the appropriate binary representation of the
C DASHLINE pattern.
C
    Locate the string at the left end, but vertically centered.
C
        CALL GSTXAL(1,3)
C
    Set the character height.
C
        CALL GSCHH(.012)
C
        ORY = ORG+.09
        GO TO ( 20, 30, 40, 50, 60), K
    20 CALL GTX (.1, ORY, 'IPAT=0001110001111111')
GO TO 70
30 CALL GTX (.1, ORY, 'IPAT=1111000011110000')
GO TO 70
40 CALL GTX (.1, ORY, 'IPAT=1111110011111100')
GO TO 70
50 CALL GTX (.1, ORY, 'IPAT=1111111100000000')
GO TO 70
60 CALL GTX (.1, ORY, 'IPAT=1111111111111100')
C
70 CONTINUE
C
CALL GSTXAL (2, 3)
CALL GTX (.5, .985, 'DEMONSTRATION PLOT FOR DASHLINE')
CALL FRAME
C
IERROR = 0
WRITE (6, 1001)
C
RETURN
1001 FORMAT (' DASHLINE TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR DASHLINE

IPAT = 0001110001111111

IPAT = 1111000011110000

IPAT = 111110011111100

IPAT = 1111110010001000

IPAT = 1111111111111100

August 1987
DASHSMTH Example

SUBROUTINE TDASHS (IERROR)
C
C PURPOSE
To provide a simple demonstration of DASHSMTH
C
C USAGE
CALL TDASHS (IERROR)
C
C ARGUMENTS
C
ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
I/O
If the test is successful, the message
DASHSMTH TEST EXECUTED--SEE PLOT TO CERTIFY
C
is printed on unit 6. In addition, 1
frame is produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plot.
C
C PRECISION
Single
C
C REQUIRED ROUTINES
DASHSMTH
C
C REQUIRED GKS LEVEL
OA
C
C LANGUAGE
FORTRAN 77
C
C ALGORITHM
TDASHS utilizes the software DASHSMTH
routines DASHDB, DASHDC, FRSTD, VECTD,
LINED, and CURVED to draw 5 curves on 1
picture using 5 different DASHSMTH patterns.
Each curve is centered about solid axis
lines and labeled with the binary
representation of the DASHSMTH pattern used.
C
X contains the abscissae and Y the ordinates of the curve to be plotted.
DIMENSION X(31) , Y(31)
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C Set a solid dash pattern. 111111111111111 (BINARY).
C Boolean operations (using locally-implemented support
C routines) are used to support porting to hosts with 16
C bit integers.
August 1987
C
ISOLID = IOR (ISHIFT (32767,1), 1)
C
DO 130 K=1,5
   CALL DASHDB (ISOLID)
   ORG = 1.07-0.195*K
C
C Draw the central axis for each curve.
C
   CALL FRSTD (.50, ORG-0.03)
   CALL VECTD (.50, ORG+0.03)
   CALL LASTD
   CALL LINED (.109, ORG,.891, ORG)
C
C Call DASHDC with a different dashed line and character combination
C for each of 5 different curves.
C
   GO TO ( 10, 20, 30, 40, 50), K
10   CALL DASHDC ('$''$' '$'$''$''$' $K = 1',10,12)
   GO TO 60
20   CALL DASHDC ('$$$$''$''$$$$$$K = 2',10,12)
   GO TO 60
30   CALL DASHDC ('$$$$''$*$$''$$$$''K = 3',10,12)
   GO TO 60
40   CALL DASHDC ('$$$$''$$''''''''''$K = 4',10,12)
   GO TO 60
50   CALL DASHDC ('$$$''$$''$$''$''$''$K = 5',10,12)
60   CONTINUE
C
C Compute the curve coordinates and draw the curve.
C
   DO 70 I=1,31
      THETA = FLOAT(I-1)*3.1415926535897932/15.
      X(I) = 0.5+.4*COS(THETA)
      Y(I) = ORG+.075*SIN(FLOAT(K)*THETA)
    70   CONTINUE
   CALL CURVED (X,Y, 31)
C
C Label the curves with the character representation of the appropriate
C DASHSMTH pattern. In the pattern labels, A and D should be interpreted
C as the apostrophe and dollar sign.
C
C Locate the string at the left end, but vertically centered.
C
   CALL GSTXAL(1,3)
C
C Set the character height.
C
   CALL GSCHH(.012)
C
   ORY = ORG+.089
   GO TO ( 80, 90, 100, 110, 120), K
80   CALL GTX(.1, ORY,'IPAT=DADADADADADADADK=1')
DASHSMTH

GO TO 130

90 CALL GTX(.1,ORY,'IPAT=DDDDDDADADDDDDDK=2')
GO TO 130

100 CALL GTX(.1,ORY,'IPAT=DDDDADDDADDDDDAK=3')
GO TO 130

110 CALL GTX(.1,ORY,'IPAT=DDDDDAAAAADDDDDK=4')
GO TO 130

120 CALL GTX(.1,ORY,'IPAT=DDDAADDADDDDDDK=5')

C
130 CONTINUE

C
CALL GSTXAL(2,3)
CALL GTX (.5,.985,'DEMONSTRATION PLOT FOR DASHSMTH')
CALL GTX (.5,.013,'IN IPAT STRINGS, A AND D SHOULD BE INTERPRETED
1AS APOSTROPHE AND DOLLAR SIGN')
CALL FRAME

C

IERROR = 0
WRITE (6,1001)
C
RETURN

C
1001 FORMAT (' DASHSMTH TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR DASHSMTH

IPAT=DADADADADADADAD=1

IPAT=DDDDDDADADADDDDDK=2

IPAT=DDDDADDDDDADDDDDAK=3

IPAT=DDDDAAAAADDDDDDK=4

IPAT=DDDAADDDADDDDDADDDK=5

IPAT STRINGS, A AND B SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SI
DASHSUPR Example

SUBROUTINE TDASHP (IERROR)
C
C PURPOSE
C
To provide a simple demonstration of DASHSUPR
C
C USAGE
C
CALL TDASHP (IERROR)
C
C ARGUMENTS
C
C
C
C
C
ON OUTPUT
C
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
If the test is successful, the message
DASHSUPR TEST EXECUTED--SEE PLOT TO CERTIFY
C
is printed on unit 6. In addition, 1 frame is produced on the machine graphics
device. In order to determine if the test was successful, it is necessary to examine
the plot.
C
C PRECISION
C
Single
C
C REQUIRED ROUTINES
C
DASHSUPR
C
C REQUIRED GKS LEVEL
C
OA
C
C LANGUAGE
C
FORTRAN 77
C
C ALGORITHM
C
TDASHP utilizes the software DASHSUPR routines DASHDB, DASHDC, FRSTD, VECTD,
LINED, and CURVED to draw 5 curves on 1 picture using 5 different DASHSUPR patterns.
Each curve is centered about solid axis lines and labeled with the binary
representation of the DASHSUPR pattern used.
C
X contains the abscissae and Y the ordinates of the curve to be plotted.
C
DIMENSION X(31), Y(31)
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C RESET initializes the model picture array. It should be called with
C each new frame. The call should occur before a call to any other
C DASHSUPR routine.
CALL RESET

C Set a solid dash pattern, 1111111111111111 (BINARY). Boolean operations (using locally-implemented support routines) are used to support porting to hosts with 16 bit integers.

ISOLID = IOR (ISHIFT (32767,1), 1)

DO 130 K=1,5
   CALL DASHDB (ISOLID)
   ORG =1.07-0.195*K
C
C Draw the central axis for each curve.
C
   CALL FRSTD (.50,ORG-0.03)
   CALL VECTD (.50,ORG+0.03)
   CALL LASTD
   CALL LINED (.109,ORG,.891,ORG)
C
C Call DASHDC with a different dashed line and character combination for each of 5 different curves.
C
   GO TO ( 10, 10 CALL DASHDC
   GO TO 60 20, 30, 40, 50),K
10 CALL DASHDC (''$''$''$''$''''$''''$''''$''$''''$K = 1',10,12)
   GO TO 60
20 CALL DASHDC (''$''$''$''$''$''$''$''$''$''$''$K = 2',10,12)
   GO TO 60
30 CALL DASHDC (''$''$''$''$''$''$''$''$''$''$''$K = 3',10,12)
   GO TO 60
40 CALL DASHDC (''$''$''$''$''$''$''$''$''$''$''$K = 4',10,12)
   GO TO 60
50 CALL DASHDC (''$''$''$''$''$''$''$''$''$''$''$K = 5',10,12)
60 CONTINUE
C
C Compute the curve coordinates and draw the curve.
C
   DO 70 I=1,31
      THETA = FLOAT(I-1)*3.1415926535897932/15.
      X(I) = 0.5+.4*COS(THETA)
      Y(I) = ORG+.075*SIN(FLOAT(K)*THETA)
70 CONTINUE
C
C Label the curves with the character representation of the appropriate DASHSUPR pattern. In the pattern labels, A and D should be interpreted as the apostrophe and dollar sign.
C
C Locate the string at the left end, but vertically centered.
C
   CALL GSTXAL(1,3)
DASHSUPR

C
C SET CHARACTER HEIGHT
C
CALL GSCHH(.012)
C
ORY = ORG+.089
GO TO ( 80, 90,100,110,120),K
80 CALL GTX(.1,ORY,'IPAT=DADADADADADADADK=1')
GO TO 130
90 CALL GTX(.1,ORY,'IPAT=DDDDDADDDDDADDDK=2')
GO TO 130
100 CALL GTX(.1,ORY,'IPAT=DDDDADDDDADDDADDK=3')
GO TO 130
110 CALL GTX(.1,ORY,'IPAT=DDDDDAAAAADDDDDK=4')
GO TO 130
120 CALL GTX(.1,ORY,'IPAT=DDDADDDADDDADDDK=5')
C
130 CONTINUE
C
CALL GSTXAL(2,3)
CALL GTX (.5,.985,'DEMONSTRATION PLOT FOR DASHSUPR')
CALL GTX (.5,.013, 'IN IPAT STRINGS, A AND D SHOULD BE INTERPRETED
1AS APOSTROPHE AND DOLLAR SIGN')
CALL FRAME
C
IERROR = 0
WRITE (6,1001)
C
RETURN
C
1001 FORMAT (' DASHSUPR TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR DASHSUPR

IPAT=DADADADADADADADK=1

---

IPAT=DDDDDDADADDDDDDDDK=2

IPAT=DDDDADDDDDDDADDK=3

IPAT=DDDDAAAAADDDDDDDDK=4

IPAT=DDDDADDDDDADDDADDK=5

IPAT STRINGS, A AND D SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SIGN.
EZMAP Example

SUBROUTINE TEZMAP (IERROR)

C
C PURPOSE
To provide a simple demonstration of
the mapping utility, EZMAP.
C
C USAGE
CALL TEZMAP (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
IERERROR
An integer variable
= 0, if the test was successful,
= 1, the test was not successful.
C
C I/O
The map dataset must be connected to a
FORTRAN unit. See the open statement in
this test deck.
C
If the test is successful, the message
EZMAP TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 10
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED ROUTINES
EZMAP, GRIDAL
C
C REQUIRED GKS LEVEL
OA
C
C ALGORITHM
TEZMAP calls routine MAPROJ once for each of
the 10 types of projections:
stereographic,
orthographic,
Lambert conformal conic with 2 standard
parallels,
Lambert equal area,
gnomonic,
azimuthal equidistant,
cylindrical equidistant,
Mercator,
Mollweide type, and
satellite view.

August 1987
C HISTORY Written October, 1976, main entry SUPMAP.
C Updated September, 1985, main entry EZMAP.
C
DIMENSION PLM1(2),PLM2(2),PLM3(2),PLM4(2)
COMMON /ERROR/ IFRAME, IERRR
C
DATA PLM1,PLM2,PLM3,PLM4 /8*0.0/
C
C Define the center of a plot title string on a square grid of size
C 0. to 1.
C
DATA TX/0.5/, TY/0.9765/
C
C Initialize the error parameter.
C
IERRR = 0
C
C Turn on the error recovery mode.
C
CALL ENTSR(IDUM,1)
C
C Frame 1 -- The stereographic projection.
C
IFRAME = 1
C
C Set the projection-type parameter.
C
CALL MAPROJ('ST',80.0,-160.0,0.0)
C
C Set the limit parameters.
C
CALL MAPSET('MA',PLM1,PLM2,PLM3,PLM4)
C
C Set the outline-dataset parameter.
C
CALL MAPSTC('OU','PS')
C
C Draw the map.
C
CALL MAPDRW
C
C Report any errors encountered.
C
IF( NERRO(IERR).NE.0) CALL RPTERR
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C Call WTSTR to write the plot title.
C
CALL WTSTR(TX,TY,'EZMAP DEMONSTRATION: STEREOGRAPHIC PROJECTION')
Frame 2 -- The orthographic projection.

IFRAME = 2

Set the projection-type parameter.

CALL MAPROJ('OR',60.0,-120.0,0.0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF( NERRO(IERR).NE.0) CALL RPTERR

Write the title.

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX,TY,'EZMAP DEMONSTRATION: ORTHOGRAPHIC PROJECTION',2,0,0)

CALL FRAME

CONTINUE
Frame 3 -- The Lambert conformal conic projection.

IFRAME = 3

Set the projection-type, limits, and outline-dataset parameters.

CALL MAPROJ('LC',45.0,-100.0,45.0)
CALL MAPSET('CO',50.0,-130.0,20.0,-75.0)
CALL MAPSTC('OU','US')

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF( NERRO(IERR).NE.0) CALL RPTERR

Select normalization transformation 0.

CALL GSELENT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX,TY,
+ 'EZMAP DEMONSTRATION: LAMBERT CONFORMAL CONIC PROJECTION',
+ 2,0,0)

CALL FRAME
30 CONTINUE
EZMAP DEMONSTRATION: LAMBERT CONFORMAL CONIC PROJECTION
Frame 4 -- The Lambert equal area projection.

IFRAME = 4

Set the projection-type, limits, and outline-dataset parameters.

CALL MAPROJ('LE', 20.0, -40.0, 0.0, 0.0)
CALL MAPSET('MA', PLM1, PLM2, PLM3, PLM4)
CALL MAPSTC('OU', 'CO')

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF ( NERRO(IERR) .NE. 0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX, TY,
+ 'EZMAP DEMONSTRATION: LAMBERT EQUAL AREA PROJECTION', 2, 0, 0)
CALL FRAME

40 CONTINUE

Important Note: EZMAP examples continue on the next 13 pages, which are numbered 478a-m. Sequential numbering resumes on page 479, where EZMAPA examples begin.
EZMAP DEMONSTRATION: LAMBERT EQUAL AREA PROJECTION
Frame 5 -- The gnomonic projection.

IFRAME = 5

Set the projection-type parameter.

CALL MAPROJ('GN', 0.0, 0.0, 0.0, 0.0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF( NERRO(IERR) .NE. 0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX,TY, 'EZMAP DEMONSTRATION: Gnomonic Projection',
+ 2,0,0)

CALL FRAME

50 CONTINUE
Frame 6 -- The azimuthal equidistant projection.

IFRAME = 6

Set the projection-type parameter.

CALL MAPROJ('AE',-20.0,40.0,0.0)

Set the grid spacing.

CALL MAPSTR('GR',5.0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF( NERRO(IERR).NE.0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX,TY,
   'EZMAP DEMONSTRATION: AZIMUTHAL EQUIDISTANT PROJECTION',
   2,0,0)

CALL FRAME

CONTINUE
EZMAP DEMONSTRATION: AZIMUTHAL EQUIDISTANT PROJECTION
Frame 7 -- The cylindrical equidistant projection.

IFRAME = 7

Set the map projection type parameter.

CALL MAPROJ('CE',-40.0,80.0,0.0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF( NERRO(IERR).NE.0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX,TY,
  + 'EZMAP DEMONSTRATION: CYLINDRICAL EQUIDISTANT PROJECTION',
  + 2,0.0)

CALL FRAME
70 CONTINUE
EZMAP DEMONSTRATION: CYLINDRICAL EQUIDISTANT PROJECTION
Frame 8 -- The mercator projection.

IFRAME = 8

Set the map projection type parameter.

CALL MAPROJ('ME',-60.0,120.0,0.0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF( NERRO(IERR).NE.0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX,TY,'EZMAP DEMONSTRATION: MERCATOR PROJECTION'
     + 2,0.0)

CALL FRAME

CONTINUE
Frame 9 -- The Mollyweide-type projection.

IFRAME = 9

Set the map projection type parameter.

CALL MAPROJ('MO', -80.0, 160.0, 0.0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF (NERRO(IERR).NE.0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX, TY, 'EZMAP DEMONSTRATION: MOLLWEIDE-TYPE PROJECTION', 2, 0, 0)

CALL FRAME

CONTINUE
EZMAP DEMONSTRATION: MOLLWEIDE-TYPE PROJECTION
Frame 10 -- The satellite view projection.

IFRAME = 10

Set the map projection type parameter.

CALL MAPROJ('SV', 0.0, -135.0, 0.0)

Set the satellite distance and suppress grid lines.

CALL MAPSTR('SA', 6.631)
CALL MAPSTI('GR', 0)

Draw the map.

CALL MAPDRW

Report any errors encountered.

IF (NERRO(IERR).NE.0) CALL RPTERR

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(TX, TY, + 'EZMAP DEMONSTRATION: SATELLITE VIEW PROJECTION', 2, 0, 0)

CALL FRAME

CONTINUE

IF (IERRR.EQ.0) WRITE(6, 1000)
IF (IERRR.EQ.1) WRITE(6, 1001)
IERROR = IERR
RETURN

1000 FORMAT('EZMAP TEST EXECUTED--SEE PLOTS TO CERTIFY')
1001 FORMAT('EZMAP TEST UNSUCCESSFUL')

END
EZMAPA Example

SUBROUTINE TEZMPA (IERROR)
C
C PURPOSE
To provide a simple demonstration of the use
of EZMAPA.
C
C USAGE
CALL TEZMPA (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
IERERROR
an error parameter
= 0, if the test is successful,
= 1, otherwise
C
C I/O
If the test is successful, the message
EZMAPA TEST SUCCESSFUL ... SEE PLOT
TO VERIFY PERFORMANCE
is written on unit 6.
C
C PRECISION
Single.
C
C REQUIRED LIBRARY
EZMAP, AREAS, SPPS
C
C FILES
OA
C
C REQUIRED GKS LEVEL
OA
C
C LANGUAGE
FORTRAN
C
C HISTORY
C
C ALGORITHM
TEZMAP draws a solid-color map of a portion
of Europe.
C
C PORTABILITY
FORTRAN 77
C
C Define an array in which to construct the area map.
C
DIMENSION IAM(25000)
C
C Dimension the arrays needed by ARSCAM and ARDRLN for x/y coordinates.
C
DIMENSION XCS(200),YCS(200)
C
C Dimension the arrays needed by ARSCAM and ARDRLN for area and group
C identifiers.
C
August 1987 479
DIMENSION IAI(2),IAG(2)
C Define the RGB color triples needed below.
C
DIMENSION RGB(3,14)
C Define a set of indices to order the colors.
C
DIMENSION IOC(14)
C Define an array for GKS aspect source flags.
C
DIMENSION IF(13)
C Declare the routine which will color the areas.
C
EXTERNAL COLRAM
C Declare the routine which will draw lines of latitude and longitude
C over water.
C
EXTERNAL COLRLN
C
DATA IOC / 6,2,5,12,10,11,1,3,4,8,9,7,13,14 /
C
DATA RGB / 0.70 , 0.70 , 0.70 ,
+ 0.75 , 0.50 , 1.00 ,
+ 0.50 , 0.00 , 1.00 ,
+ 0.00 , 0.00 , 1.00 ,
+ 0.00 , 0.50 , 1.00 ,
+ 0.00 , 1.00 , 1.00 ,
+ 0.00 , 1.00 , 0.60 ,
+ 0.00 , 1.00 , 0.00 ,
+ 0.70 , 1.00 , 0.00 ,
+ 1.00 , 1.00 , 0.00 ,
+ 1.00 , 0.75 , 0.00 ,
+ 1.00 , 0.38 , 0.38 ,
+ 1.00 , 0.00 , 0.38 ,
+ 1.00 , 0.00 , 0.00 /
C
C Set the aspect source flags for FILL AREA INTERIOR STYLE and for
C FILL AREA STYLE INDEX to "individual".
C
CALL GQASF (IE,IF)
IF(11)=1
IF(12)=1
CALL GSASF (IF)
C Force solid fill.
C
CALL GSFAIS (1)
C Define 15 different color indices. The first 14 are spaced through
C the color spectrum and the final one is black.
C
CALL SETUSV ('IM',15)
C
DO 101 J=1,14
I=IOC(J)
IF (RGB(1,I).GE.RGB(2,I).AND.RGB(1,I).GE.RGB(3,I)) THEN
  IND=1
  IND=2
ELSE IF (RGB(3,I).GE.RGB(1,I).AND.RGB(3,I).GE.RGB(2,I)) THEN
  IND=3
END IF
CALL SETUSV ('IR',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
CALL SETUSV ('IG',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
CALL SETUSV ('IB',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
101 CONTINUE
C
CALL SETUSV ('IR',1)
CALL SETUSV ('IG',1)
CALL SETUSV ('IB',1)
CALL SETUSV ('IN',0)
C
C Set up EZMAP.
C
CALL MAPSTC ('OU','PO')
CALL MAPROJ ('ME',0.,0.,0.)
CALL MAPSET ('CO',30.,-15.,60.,30.)
C
C Make MAPBLA use 1 and 2 as the group identifiers.
C
CALL MAPSTI ('G1',1)
CALL MAPSTI ('G2',2)
C
C Use 5 vertical strips to reduce the number of points defining the
C sub-areas.
C
CALL MAPSTI ('VS',5)
C
C Initialize EZMAP.
C
CALL MAPINT
C
C Initialize the area map.
C
CALL ARINAM (IAM,25000)
C
C Add edges to the area map.
C
CALL MAPBLA (IAM)
C
C Pre-process the area map.
CALL ARPRAM (IAM, 0, 0, 0)

C Compute and print the amount of space used in the area map.

ISU=25000-(IAM(6)-IAM(5)-1)
PRINT *, 'SPACE USED IN AREA MAP IS ',ISU

C Color the map.

CALL ARSCAM (IAM, XCS, YCS, 200, IAI, IAG, 2, COLRAM)

C In black, draw a perimeter and outline all the countries.

CALL SETUSV ('II',15)
CALL MAPSTI ('LA',0)
CALL MAPSTI ('MV',1)
CALL MAPLBL
CALL MAPLOT

C Draw lines of latitude and longitude over water.

CALL MAPGRM (IAM, XCS, YCS, 200, IAI, IAG, 2, COLRLN)

C Advance the frame.

CALL FRAME

C Done.

IERROR=0
WRITE (6,1001)
RETURN

1001 FORMAT (' EZNAPA TEST SUCCESSFUL', 24X,
              'SEE PLOT TO VERIFY PERFORMANCE')

END

SUBROUTINE COLRAM (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
ITM=1
DO 101 I=1,NAI
   IF (IAI(I).LT.0) ITM=0
101 CONTINUE
   IF (ITM.NE.0) THEN
      ITM=0
      DO 102 I=1,NAI
         IF (IAG(I).EQ.1) ITM=IAI(I)
102 CONTINUE
   IF (ITM.GT.0) THEN
      IF (NCS.GT.150) PRINT * , 'COLRAM - NCS TOO BIG - ',NCS
      CALL SETUSV ('II', MAPACI(ITM))
CALL GFA (NCS-1,XCS,YCS)
END IF
END IF
RETURN
END
SUBROUTINE COLRLN (XCS,YCS,NCS,IAI, IAG, NAI)
DIMENSION XCS(*),YCS(*),IAI(*),IAG(*)
ITM=1
DO 101 I=1,NAI
   IF (IAI(I).LT.0) ITM=0
101 CONTINUE
IF (ITM.NE.0) THEN
   ITM=0
   DO 102 I=1,NAI
      IF (IAG(I).EQ.1) ITM=IAI(I)
102 CONTINUE
   IF (MAPACI(ITM).EQ.1) THEN
      IF (NCS.GT.150) PRINT *
      'COLRLN - NCS TOO BIG - ',NCS
      CALL SETUSV ('II',15)
      CALL GPL (NCS,XCS,YCS)
   END IF
END IF
RETURN
END

Important Note: The code above produces a solid-colored map of Europe, the United Kingdom, northern Africa and eastern Asia if the output is directed to a color output device. The plot published here was produced on a black-and-white device, using hollow fill rather than solid fill. Obviously, it is unsatisfactory; to get a satisfactory plot requires using a color device with solid fill. (On devices with solid fill, but no color, you may get just a solid square. On devices with color, but hollow fill, you may get the same plot, but with the lines in color.)

Even though the black-and-white plot is unsatisfactory, it is reproduced here because the plot shows all of the edges of all of the polygonal areas that are colored to produce the desired plot. The four vertical lines splitting the plot into five equal vertical strips are introduced by the EZMAP edge group 'G2'. Other vertical lines are introduced by AREAS in order to define polygons with holes.

Longitude and latitude lines also appear on the plot reproduced here, but you should not confuse them with the lines that define edges. The longitude and latitude lines appear only in water and are hidden over land masses. In contrast, the lines that define edges are shown all the way across the plot.
GRIDAL Examples

The following FORTRAN code provides a sample test driver for the GRIDAL utility of NCAR graphics. The plots for calls to the entries GRID, GRIDL, PERIM, PERIML, HALFAX, TICK4, LABMOD, and GRIDAL appear directly following the code that generates them.

SUBROUTINE TGRIDA(IER)

C
C PURPOSE
C
C To provide a simple demonstration of all of the entry points of the GRIDAL package. Eight plots are produced.
C
C USAGE
C
C CALL TGRIDA (IERERROR)
C
C ARGUMENTS
C
C IERROR
C
C An integer variable
C
C = 0, if the test was successful,
C
C = 1, otherwise
C
C I/O
C
C If the test is successful, the message
C
C GRIDAL TEST EXECUTED--SEE PLOTS TO CERTIFY
C
C is printed on unit 6. In addition, 8 frames are produced on the machine graphics device. In order to determine if the test was successful, it is necessary to examine these plots.
C
C PRECISION
C
C Single
C
C LANGUAGE
C
C FORTRAN 77
C
C ALGORITHM
C
C All of the entries of the GRIDAL package are invoked (GRID, GRIDL, PERIM, PERIML, HALFAX, TICK4, LABMOD, and GRIDAL) to produce plots. The GRIDAL entry is called ten times. The first call is to demonstrate a full frame grid. The next nine calls create a single frame that contains the nine legal IGPH grid options to show how up to nine plots can be placed on a single frame.

CHARACTER*2 BUFF
C
C Define normalization transformation
August 1987
The following code and associated plot show how a simple grid is formed.

```
CALL GSWN(1.0.,1.0.,1.)
CALL GSVP(1.0.,2.0.,8.0.,8.)

C GRID
C
CALL GSELNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR GRID',2,0,0)
CALL GSELNT(1)
CALL GRID(5,2,6,3)
CALL FRAME

DEMONSTRATION PLOT FOR GRID
```
The following code and associated plot show a grid and labels.

```
C
C GRIDL
C
CALL GSELTNT(0)
CALL WTSTR(0.5, 0.9, 'DEMONSTRATION PLOT FOR GRIDL', 2, 0.0)
CALL GSELTNT(1)
CALL GRIDL(5, 2, 6, 3)
CALL FRAME
```

DEMONSTRATION PLOT FOR GRIDL

```
+-------------------+
|                   |
|                   |
|                   |
|                   |
|                   |
|                   |
+-------------------+
```

August 1987
The following code and associated plot show how a simple perimeter is formed.

```c
C
C PERIM
C
CALL GSELNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR PERIM',2,0,0)
CALL GSELNT(1)
CALL PERIM(5,2,6,3)
CALL FRAME

DEMONSTRATION PLOT FOR PERIM
```
The following code and associated plot show how a labeled perimeter is formed.

C
C PERIML
C
CALL GSELNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR PERIML',2,0,0)
CALL GSELNT(1)
CALL PERIML(5,2,6,3)
CALL FRAME

DEMONSTRATION PLOT FOR PERIML

August 1987
The following code and associated plot show how a half axis is formed.

C
C HALFAX
C
CALL GSELNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR HALFAX',2,0,0)
CALL GSELNT(1)
CALL HALFAEX(5,2,6,.3,.5,0,0)
CALL FRAME

DEMONSTRATION PLOT FOR HALFAX
The following code and associated plot show how tick marks are formed.

```
C
C TICK4
C
CALL GSELNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR TICK4',2,0,0)
CALL GSELNT(1)
CALL TICK4(150,50,150,50)
CALL PERIM(5,2,6,3)
CALL FRAME
CALL TICK4(12,8,12,8)
```
The following code and associated plot show how LABMOD can be used to change the formats of the printed labels.

```
C
C LABMOD
C
CALL GSELNT(0)
CALL WTSTR(.5, .9, 'DEMONSTRATION PLOT FOR LABMOD', 2, 0, 0)
CALL GSELNT(1)
CALL LABMOD(' (E10.2) ', (F4.2) ', 10, 4, 15, 15, 0, 0, 0)
CALL HALFAX(2, 1, 10, 1, 0., 0., 1, 1)
CALL FRAME
C
C Use LABMOD to reduce the number of digits in the grid scales
C
```

DEMONSTRATION PLOT FOR LABMOD
The following code and associated plot show the creation of a grid background using the subroutine GRIDAL.

CALL LABMOD(' (F4.2)', '(F4.2)', 4, 4, 0, 0, 0, 0, 0, 0)
C
C GRIDAL - A single grid on a frame.
C
IGPH = 0
WRITE (BUFF, 1001) IGPH
CALL GSELNT(0)
CALL WTSTR(.5, .85, 'IGPH = ', 2, 0, 1)
CALL WTSTR(.5, .85, BUFF, 2, 0, -1)
CALL WTSTR(.5, .9, 'DEMONSTRATION PLOT FOR GRIDAL', 2, 0, 0)
CALL GSELNT(1)
CALL GRIDAL(5, 2, 6, 3, 1, 1, IGPH, .3, .13)
CALL FRAME

DEMONSTRATION PLOT FOR GRIDAL
IGPH = 0

August 1987
The following code and associated plots show the nine kinds of grid line options which are available in subroutine GRIDAL.

```
C
C GRIDAL - All 9 legal grids on a single frame.
C
CALL GSELNT(0)
CALL WTSTR(.5,.98,'TEST IGPH OPTIONS OF GRIDAL',2,0,0)
KNT = 0
DO 100 I=0,10
   IF (I .EQ. 3 .OR. I .EQ. 7) GOTO 100
   IGPH = I
   WRITE (BUFF,1001) IGPH
   KNT = KNT + 1
C
C Compute the X and Y grid boundaries for the 9 plots.
C
   Y1 = .42
   IF(KNT.LT.4) Y1 = .74
   IF(KNT.GT.6) Y1 = .10
   X1 = .10
   IF(KNT.EQ.2.OR.KNT.EQ.5.OR.KNT.EQ.8) X1 = .42
   IF(KNT.EQ.3.OR.KNT.EQ.6.OR.KNT.EQ.9) X1 = .74
   X2 = X1 + .18
   Y2 = Y1 + .18
C
C Specify some user coordinates.
C
   H1 = X1*10.
   H2 = X2*10.
   V1 = Y1*10.
   V2 = Y2*10.
C
C Locate the IGPH legend above the grid.
C
   XG = X1 + .13
   YG = Y2 + .03
   CALL GSELNT(0)
   CALL WTSTR(XG,YG,'IGPH = ',2,0,1)
   CALL WTSTR(XG,YG,BUFF,2,0,-1)
   CALL GSELNT(1)
   CALL SET(X1,X2,Y1,Y2,H1,H2,V1,V2,1)
   CALL GRIDAL(3,3,4,2,1,1,IGPH,H1,V1)
100 CONTINUE
C
C Advance the frame.
C
   CALL FRAME
1001 FORMAT(I2)
C
   WRITE (6,600)
600 FORMAT(' GRIDAL TEST EXECUTED--SEE PLOTS TO CERTIFY')
```
GRIDAL

RETURN
END

TEST IGPH OPTIONS OF GRIDAL

<table>
<thead>
<tr>
<th>IGPH = 0</th>
<th>IGPH = 1</th>
<th>IGPH = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.20</td>
<td>9.20</td>
<td>9.20</td>
</tr>
<tr>
<td>8.75</td>
<td>8.75</td>
<td>8.75</td>
</tr>
<tr>
<td>8.30</td>
<td>8.30</td>
<td>8.30</td>
</tr>
<tr>
<td>7.85</td>
<td>7.85</td>
<td>7.85</td>
</tr>
<tr>
<td>7.40</td>
<td>7.40</td>
<td>7.40</td>
</tr>
<tr>
<td>1.00 1.60 2.20 2.80</td>
<td>4.20 4.80 5.40 6.00</td>
<td>7.40 8.00 8.60 9.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IGPH = 4</th>
<th>IGPH = 5</th>
<th>IGPH = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.00</td>
<td>6.00</td>
<td>6.00</td>
</tr>
<tr>
<td>5.55</td>
<td>5.55</td>
<td>5.55</td>
</tr>
<tr>
<td>5.10</td>
<td>5.10</td>
<td>5.10</td>
</tr>
<tr>
<td>4.65</td>
<td>4.65</td>
<td>4.65</td>
</tr>
<tr>
<td>4.20</td>
<td>4.20</td>
<td>4.20</td>
</tr>
<tr>
<td>1.00 1.60 2.20 2.80</td>
<td>4.20 4.80 5.40 6.00</td>
<td>7.40 8.00 8.60 9.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IGPH = 8</th>
<th>IGPH = 9</th>
<th>IGPH = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.80</td>
<td>2.80</td>
<td>2.80</td>
</tr>
<tr>
<td>2.35</td>
<td>2.35</td>
<td>2.35</td>
</tr>
<tr>
<td>1.90</td>
<td>1.90</td>
<td>1.90</td>
</tr>
<tr>
<td>1.45</td>
<td>1.45</td>
<td>1.45</td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1.00 1.60 2.20 2.80</td>
<td>4.20 4.80 5.40 6.00</td>
<td>7.40 8.00 8.60 9.20</td>
</tr>
</tbody>
</table>

August 1987
HAFTON Example

SUBROUTINE THAFTO (IERROR)

C
C PURPOSE
C Provides a simple demonstration of HAFTON
C
C USAGE
C CALL THAFTO (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
C IERROR
An integer variable
= 0, if the test was successful,
= 1, the test was not successful.
C
C I/O
If the test is successful, the message
HAFTON TEST EXECUTED--SEE PLOTS TO CERTIFY
C
is printed on unit 6. In addition, 2 halftone
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.

C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED ROUTINES
HAFTON
C
C REQUIRED GKS LEVEL
OA
C
C ALGORITHM
The function
Z(X,Y) = X + Y + 1./((X-.1)**2+Y**2+.09)
-1./((X+.1)**2+Y**2+.09)
C
for X = -1. to +1. in increments of .1, and
Y = -1.2 to +1.2 in increments of .1,
is computed. Then, entries EZHFTN and HAFTON
are called to generate 2 halftone plots of Z.
C
Z contains the values to be plotted.
C
REAL Z(21,25)
SAVE
C
C Specify coordinates for plot titles. The values TX and TY
C define the center of the title string in a 0. to 1. range.
C
August 1987 497
DATA TX/0.0762/, TY/0.9769/
C Specify low (FLO) and high (FHI) contour values, and NLEV unique contour levels. NOPT determines how Z maps onto the intensities, and the directness of the mapping.
C
DATA FLO/-4.0/, FHI/4.0/, NLEV/8/, NOPT/-3/
C Initialize the error indicator
C IERROR = 0
C Fill the 2 dimensional array to be plotted
C
DO 20 I=1,21
  X = .1*FLOAT(I-11)
DO 10 J=1,25
  Y = .1*FLOAT(J-13)
  Z(I,J) = X+Y+1./((X-.10)**2+Y**2+.09) - 1.*/((X+.10)**2+Y**2+.09)
10 CONTINUE
20 CONTINUE
C Select normalization trans 0 for plotting title.
C CALL GSELNT (0)
C Frame 1 -- The EZHFTN entry with default parameters.
C Call WTSTR to write the plot title.
C CALL WTSTR (TX,TY,
  1 'DEMONSTRATION PLOT FOR ENTRY EZHFTN OF HAFTON',2,0,-1)
C Entry EZHFTN requires only the array and its dimensions.
C CALL EZHFTN (Z,21,25)
CALL FRAME
C Frame 2 -- The HAFTON entry with user selectable parameters.
C Add a plot title.
C CALL WTSTR (TX,TY,
  1 'DEMONSTRATION PLOT FOR ENTRY HAFTON OF HAFTON',2,0,-1)
C Entry HAFTON allows user specification of plot parameters.
C CALL HAFTON (Z,21,21,25,FLO,FHI,NLEV,NOPT,0,0,0.)
CALL FRAME
C
WRITE (6,1001)
RETURN
C
1001 FORMAT (' HAFTON TEST EXECUTED--SEE PLOTS TO CERTIFY')
C
END

Important Note: HAFTON output is poor on most laser printers because the dots are so small and faint that the gradations between the shades of gray are not distinct. The first two examples shown on the following pages were printed on a laser printer so that you can see the resulting problem. If you can send the HAFTON output to microfilm at your location, the examples will resemble the last two, which show the gray gradations better.
DEMONSTRATION PLOT FOR ENTRY HAFTON OF HAFTON
DEMONSTRATION PLOT FOR ENTRY EZHFTN OF HAFTON
DEMONSTRATION PLOT FOR ENTRY HAFTON OF HAFTON

August 1987

1.145
-0.2895
-1.371
-2.229
-2.914
-3.444
-3.819
HISTGR Example

The following FORTRAN code provides a sample test driver for the HISTGR utility of NCAR graphics. Three frames are generated with various combinations of options. The third frame contains four histogram plots, one in each frame quadrant.

SUBROUTINE THSTGR (IERROR)
C
C PURPOSE
C
C To provide a demonstration of the HISTGR utility and to test each of the four IFLAG options.
C
C USAGE
C
CALL THSTGR (IERROR)
C
C ARGUMENTS
C
IERROR
An error flag
= 0, If the test is successful,
= 1, otherwise.
C
I/O
If the test is successful, the message
HISTGR TEST SUCCESSFUL . . . SEE PLOT TO VERIFY PERFORMANCE
is written on unit 6.

In addition, three (3) labeled frames are produced. To determine if the test was successful examine these plots.

C PRECISION
Single
C
C REQUIRED LIBRARY
HISTGR
C
C FILES

C REQUIRED CKS LEVEL
OA
C
C LANGUAGE
FORTRAN 77
C
C HISTORY
Originally written May, 1985; revised August, 1987
C
C ALGORITHM
THSTGR computes data and calls HISTGR 5 times, exercising different options.
C
Array DAT1 is filled with values to be used as input for HISTGR
C
August 1987
PARAMETER ( NDIM=320, NCLS=17, NWRK=374 )

C
C NWRK = NDIM + 3*(NCLS+1)
C
CHARACTER*55 MON
REAL DAT1(NDIM,2), X, ARR7(7), WORK(NWRK)
REAL RGB(3,15), CLASS(NCLS+1)
REAL SPAC(2)
INTEGER IOC(15), COLORS(15)

C
C Define the RGB triples needed below.
C
DATA RGB / 0.70, 0.70, 0.70,
- 0.75, 0.00, 1.00,
- 0.30, 0.10, 1.00,
- 0.10, 0.50, 1.00,
- 0.00, 0.70, 1.00,
- 0.00, 1.00, 1.00,
- 0.00, 1.00, 0.70,
- 0.00, 1.00, 0.00,
- 0.70, 0.00, 0.70,
- 1.00, 1.00, 0.00,
- 1.00, 0.75, 0.00,
- 1.00, 0.48, 0.00,
- 1.00, 0.00, 0.48,
- 1.00, 0.00, 0.00,
- 1.00, 1.00, 1.00 /

C
DATA TX / .5/, TY / .9765 /

C
C Set maximum color index
C
CALL SETUSV ('IM', 15)

C
C Define 15 color indices, 14 spaced throughout the color
C spectrum, and the last one being white.
C
DO 100 I = 1,15
  IF (RGB(1,I).
   GE.RGB(2,I).AND.RGB(1,I).
   GE.RGB(3,I)) THEN
    IND = 1
  ELSE IF (RGB(2,I).
   GE.RGB(1,I).AND.RGB(2,I).
   GE.RGB(3,I)) THEN
    IND = 2
  ELSE IF (RGB(3,I).
   GE.RGB(1,I).AND.RGB(3,I).
   GE.RGB(2,I)) THEN
    IND = 3
  ENDIF
  CALL SETUSV ('IR', IFIX(10000.*RGB(1,I)/RGB(IND,I)))
  CALL SETUSV ('IG', IFIX(10000.*RGB(2,I)/RGB(IND,I)))
  CALL SETUSV ('IB', IFIX(10000.*RGB(3,I)/RGB(IND,I)))
  CALL SETUSV ('IN', IFIX(10000.*RGB(IND,I)))
100 CONTINUE

C
C Call ENTSR (from PORT library) to recover from warnings
C
CALL ENTSR(IDUM, 1)

C Frame 1: Demonstrate the default version of HISTGR, IFLAG = 0.
C
   IFLAG = 0
   NCLASS = 11
   NPTS = 320
C
   DO 105 I=1,NPTS
       DO 105 J=1,2
           DAT1(I,J)=0.
     105 CONTINUE
   DO 110 I=1,NPTS
       X = FLOAT(I)
       DAT1(I,1) = 10. * ALOG10(0.1*X+1.)
     110 CONTINUE
C
   (First call HSTOPL('DEF=ON') to activate all default options.)
C
   CALL HSTOPL('DE=ON')
C
   CALL WTSTR (TX,TY,
       +'DEMONSTRATION PLOT FOR DEFAULT VERSION of HISTGR',2,0,0)
   CALL HISTGR(DAT1, NDIM, NPTS, IFLAG, CLASS, NCLASS, WORK, NWRK)
DEMONSTRATION PLOT FOR DEFAULT VERSION of HISTGR
Frame 2: Demonstrate the comparison of two previously determined histograms, IFLAG = 3.

IFLAG = 3
NCLASS = 6
NPTS2 = 6
DO 200 I=1,NPTS2
  DAT1(I,1)=2*SIN(FLOAT(I))
  DAT1(I,2)=2.5*COS(FLOAT(I)/.5)
  CLASS(I)=FLOAT(I)
200 CONTINUE

(First call HSTOPL('DEF=ON') to activate all default options.)

CALL HSTOPL('DE=ON')

Turn on color, title, perimeter, frequency, format, character, label, and spacing options.

COLORS(1) = 8
COLORS(2) = 3
COLORS(3) = 14
COLORS(4) = 11
COLORS(5) = 6
COLORS(6) = 13
COLORS(7) = 14
COLORS(8) = 5
CALL HSTOPI('COL=ON',3,0,COLORS,8)

Choose large, horizontal alphanumeric labels for class labels.

CALL HSTOPC('TI=ON', +'OPTIONS CHANGED: PRM,TIT,CHA,LAB,FOR,COL,FQN and SPA',7,3)
CALL HSTOPL('FR=ON')
CALL HSTOPC('FQ=ON','MONTHLY PRECIPITATION',7,3)
CALL HSTOPC('FO=ON','(F3.0)',9,3)
MON='JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC'
CALL HSTOPL('CH=ON',MON,12,3)
CALL HSTOPC('LA=ON','COMPARE TWO DATASETS',7,3)
SPAC(1) = 2.0
SPAC(2) = -1.5
CALL HSTOPR('SP=ON',SPAC,2)

The second argument must be the actual dimension size of DAT1.

CALL HISTGR(DAT1, NDIM, NPTS2, IFLAG, CLASS, NCLASS, WORK, NWRK)

**Important Note:** The following plot is reproduced here in black and white. If the code were used to drive a color output device, the output would be in color. The bars would be green and blue, the labels pink, the heading red, and the axes gold — all surrounded by a blue box.
Frame 3: Put four plots on 1 frame by setting FRAME = OFF for the first 3 plots and FRAME = ON for the last plot.

Plot 1: IFLAG = 0, automatic sorting of the input data into NCLASS = 17 bins.

IFLAG = 0
NCLASS = 17
NPTS = 320
DO 210 I=1,NPTS
   X = FLOAT(I)
   DAT1(I,1) = 10. * ALOG10(0.1*X+1.)
210 CONTINUE

C (First call HSTOPL('DEF=ON') to activate all default options.)
CALL HSTOPL('DE=ON')

Turn on horizontal bars, title, median, window, color, and spacing options.

Choose large horizontal class labels.

CALL HSTOPL('HO=ON')
CALL HSTOPL('ME=ON')

Plot 1 goes into the top left quadrant of the frame.

ARR7 coordinates are XMIN, XMAX, YMIN, YMAX.

ARR7(1) = 0.
ARR7(2) = .5
ARR7(3) = .5
ARR7(4) = 1.

CALL HSTOPL('WI=ON',ARR7,4)

COLORS(1) = 8
COLORS(2) = 2
COLORS(3) = 10
COLORS(4) = 4
COLORS(5) = 5
COLORS(6) = 6
COLORS(7) = 7
COLORS(8) = 8

CALL HSTOPL('COL=ON',2,0,COLORS,8)

SPAC(1) = 3.0
SPAC(2) = 0.0

CALL HSTOPL('SP=ON',SPAC,2)

Turn off the frame advance.

CALL HSTOPL('FR=OF')
CALL HISTGR(DAT1, NDIM, NPTS, IFLAG, CLASS, NCLASS, WORK, NWRK)

Plot 2: IFLAG = 2, one set of 11 histogram classes and their associated values are plotted.

IFLAG = 2
NCLASS = 11
NPTS2 = 11
DO 650 I = 1, NPTS2
   CLASS(I) = FLOAT(2*I)
   DAT1(I,1) = (FLOAT(2*I))**0.5
650 CONTINUE

(First call HSTOPL('DEF=ON') to activate all default options.)
CALL HSTOPL('DE=ON')

Turn on title, label, frequency, format, and window options.

Choose medium sized, horizontal labels for class labels

CALL HSTOPC('TI=ON', + 'OPTIONS CHANGED: LAB,FQN,TIT,FOR,FRA AND WIN',11,3)
CALL HSTOPC('LA=ON', 'Y VALUES ASSIGNED TO X VALUES', 11,3)
CALL HSTOPC('FQ=ON', 'SQUARE ROOT OF CLASS MID-VALUES', &12,3)
CALL HSTOPC('FO=ON', '(I3)', 11,3)

Plot 2 goes into the top right quadrant of the frame.

ARR7(1) = .5
ARR7(2) = 1.
ARR7(3) = .5
ARR7(4) = 1.
CALL HSTOPR('WIN=ON',ARR7,4)

Turn off color and frame advance options.

CALL HSTOPI('COL=OFF',2,0,COLORS,8)
CALL HSTOPL('FR=OFF')
CALL HISTGR(DAT1, NDIM, NPTS2, IFLAG, CLASS, NCLASS, WORK, NWRK)

Plot 3: IFLAG = 1, input values are sorted into a defined set of 8 classes.

IFLAG = 1
NCLASS = 8
NPTS = 320
X = 0.
DO 500 I = 1,NPTS
   DAT1(I,1) = SIN(X)
   X = X + .02
500 CONTINUE

CLASS(1) = -0.6
DO 550 I = 1,NCLASS
   CLASS(I+1) = CLASS(I) + 0.20
550 CONTINUE

(First call HSTOPL('DEF=ON') to activate all default options.)

CALL HSTOPL('DE=ON')

Turn on class, draw line, format, title, and label options.

CALL HSTOPI('CL=ON',2,30,COLORS,8)
CALL HSTOPL('DR=ON')
CALL HSTOPC('FOR=ON','(F6.2)',9,3)
CALL HSTOPC('TI=ON',
+ 'OPTIONS CHANGED: CLA,DRL,FOR,TIT,LAB,MID,WIN,SHA and FRA',9,3)
CALL HSTOPC('LA=ON','CLASS VALUES CHOSEN FROM -0.6 to 1.0', &9,3)

C

C Plot 3 goes into the lower left quadrant of the frame.
C
ARR7(1) = 0.
ARR7(2) = .5
ARR7(3) = 0.
ARR7(4) = .5
CALL HSTOPR('WI=ON',ARR7,4)

C

C Turn off color, midvalues, frame advance and shading options.
C
CALL HSTOPI('CO=OF',2,30,COLORS,8)
CALL HSTOPL('MI=OFF')
CALL HSTOPL('FR=OFF')
CALL HSTOPL('SH=OFF')
CALL HISTGR(DAT1,NDIM,NPTS,IFLAG,CLASS,NCLASS,WORK,NWRK)

C

C Plot 4: IFLAG = 0, input values are sorted into 11 equally sized bins over the range of the input values.
C
IFLAG = 0
NCLASS = 11
NPTS = 320
X = 0.
DO 310 I = 1,NPTS
  DAT1(I,1) = SIN(X)
  X = X + .02
310 CONTINUE

C

C (First call HSTOPL('DEF=ON') to activate all default options.)
C
CALL HSTOPL('DE=ON')

C

C Turn on class, frequency, format, title, window, and label options.
C
Choose medium sized vertical class value labels.

CALL HSTOPI('CL=ON',2,90,COLORS,8)
CALL HSTOPC('FQN=ON','NUMBER OF OCCURENCES IN EACH CLASS',9,3)
CALL HSTOPC('FOR=ON','(F6.2)',9,3)
CALL HSTOPC('TI=ON',
+ 'OPTIONS CHANGED: CLA,LAB,FQN,TIT,SPA,PER,FOR AND WIN',9,3)

C

C Plot 4 goes into the lower right quadrant of the frame.
C
ARR7(1) = .5
ARR7(2) = 1.
ARR7(3) = 0.
ARR7(4) = .5
CALL HSTOPR('WIN=ON',ARR7,4)
CALL HSTOPC('LAB=ON','CLASS VALUES COMPUTED WITHIN HISTGR', &9,3)

C
C Turn off color, spacing and percent axis options.
C
CALL HSTOPI('CO=OF',2,90,_COLORS,8)
CALL HSTOPR('SP=OF',SPAC,2)
CALL HSTOPL('PER=OFF')

C
CALL HISTGR(DAT1, NDIM, NPTS, IFLAG, CLASS, NCLASS, WORK, NWRK)

C
IERRO R = 0
WRITE (6,1001)
C
RETURN

C
1001 FORMAT ('HISTGR TEST SUCCESSFUL',24X,
1 'SEE PLOT TO VERIFY PERFORMANCE')
END

Important Note: The following example plots are reproduced here in black and white. The upper left example would be in color if it were produced on a color output device. The bars would be green and blue, and other parts of the example would also be in color.
August 1987
515
SUBROUTINE TISOSR (IERROR)
C
C PURPOSE
C To provide a simple demonstration of ISOSRF.
C
C USAGE
C CALL TISOSR (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
C IERROR
An integer variable
= 0, if the test was successful,
= 1, the test was not successful.
C
C I/O
If the test is successful, the message
ISOSRF TEST EXECUTED--SEE PLOTS TO CERTIFY
C
is printed on unit 6. In addition, 2
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED ROUTINES
ISOSRF
C
C REQUIRED GKS LEVEL
OA
C
C ALGORITHM
Values of a function on a 3-D rectangular grid
are stored in array T. Entries EZISOS and
ISOSRF are called to draw iso-surfaced
plots of the function.
C
SAVE
DIMENSION T(21, 31, 19), SLAB(33, 33), EYE(3)
C
Specify coordinates for plot titles.
C
REAL IX, IY
DATA IX/.44/, IY/.95/
C
DATA NU, NV, NW/21, 31, 19/
DATA RBIG1, RBIG2, RSML1, RSML2/6., 6., 2., 2./
DATA TISO/0./
DATA MUVWP2/33/
DATA IFLAG/-7/
C Initialize the error parameter.
C
IERROR = 1
C
C Fill the 3-D array to be plotted.
C
JCENT1 = FLOAT(NV)*.5-RBIG1*.5
JCENT2 = FLOAT(NV)*.5+RBIG2*.5
DO 30 I=1,NU
   FIMID = I-NU/2
   DO 20 J=1,NV
      FJMID1 = J-JCENT1
      FJMID2 = J-JCENT2
      DO 10 K=1,NW
         FKMID = K-NW/2
         F1 = SQRT(RBIG1*RBIG1/(FJMID1*FJMID1+FKMID*FKMID+.1))
         F2 = SQRT(RBIG2*RBIG2/(FIMID*FIMID+FJMID2*FJMID2+.1))
         FIP1 = (1.-F1)*FIMID
         FIP2 = (1.-F2)*FIMID
         FJP1 = (1.-F1)*FJMID1
         FJP2 = (1.-F2)*FJMID2
         FKP1 = (1.-F1)*FKMID
         FKP2 = (1.-F2)*FKMID
         T(I,J,K) = AMIN1(FIMID*FIMID+FJP1*FJP1+FKP1*FKP1-
                        RSML1*RSML1,
                        FKMID*FKMID+FIP2*FIP2+FJP2*FJP2-RSML2*RSML2)
   10 CONTINUE
20 CONTINUE
30 CONTINUE
C
C Define the eye position.
C
EYE(1) = 100.
EYE(2) = 150.
EYE(3) = 125.
C
C Frame 1 -- The EZISOS entry.
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C Call WTSTR to write the plot title.
C
CALL WTSTR(IX,IY,'DEMONSTRATION PLOT FOR ENTRY EZISOS OF ISOSRF',
            1,2,0,0)
CALL EZISOS (T,NU,NV,NW,EYE,SLAB,TISO)
DEMONSTRATION PLOT FOR ENTRY EZISOS OF ISOSRF

August 1987

ISOSRF

519
Frame 2 -- The ISOSRF entry.

Select normalization transformation 0.

CALL GSELNT(0)

Call WTSTR to write the plot title.

CALL WTSTR(IX, IY, 'DEMONSTRATION PLOT FOR ENTRY ISOSRF OF ISOSRF', 1 2, 0, 0)

Test ISOSRF with a subarray of T.

MU = NU/2
MV = NV/2
MW = NW/2
MUVWP2 = MAXO(MU, MV, MW) + 2
CALL ISOSRF(T(MU, MV, MW), NU, MU, NV, MV, MW, EYE, MUVWP2, SLAB, TISO, IFLAG)
CALL FRAME

IERRO = 0
WRITE (6, 1001)
RETURN

1001 FORMAT ('ISOSRF TEST EXECUTED--SEE PLOTS TO CERTIFY')

END
DEMONSTRATION PLOT FOR ENTRY ISOSRF OF ISOSRF
August 1987
ISOSRFHR Example

SUBROUTINE TISOHR (IERROR)

C PURPOSE
To provide a simple demonstration of ISOSRFHR.

C USAGE
CALL TISOHR (IERROR)

C ARGUMENTS
C
C IERROR
An integer variable
= 0, if the test was successful,
= 1, the test was not successful.

C I/O
A scratch file must be assigned to unit IUNIT.
Common block UNITS should be inserted in the
calling program. Then, set IUNIT in the
calling program.

If the test is successful, the message

ISOSRFHR TEST EXECUTED--SEE PLOT TO CERTIFY

is printed on unit 6. In addition, 1
frame is produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plot.

C PRECISION
Single

C LANGUAGE
FORTRAN 77

C REQUIRED ROUTINES
ISOSRFHR

C REQUIRED GKS LEVEL
QA

C ALGORITHM
This test program draws a perspective view
of 2 interlocking doughnuts.

DIMENSION EYE(3), S(4), IS2(4,200),
1 ST1(81,51,2), IOBJS(81,51)
COMMON /UNITS/ IUNIT
SAVE

Specify coordinates for plot titles on a square grid of integer
coordinates that range from 1 to 1024. IX and IY define the center
of the title string.

DATA IX/448/, IY/990/

August 1987
C Define the eye position.
C
DATA EYE(1), EYE(2), EYE(3) / 200., 250., 250. /
C
C Define the overall dimensions of the box containing the objects.
C
DATA NU, NV, NW / 51, 81, 51 /
C
C Specify the dimensions of the model of the image plane.
C
DATA LX, NX, NY / 4., 180., 180 /
C
C Specify the user viewport in a 1 to 1024 range.
C
DATA S(1), S(2), S(3), S(4) / 10., 1010., 10., 1010. /
DATA MV / 81 /
C
C Specify the large and small radii for the individual doughnuts.
C
DATA RBIG1, RBIG2, RSML1, RSML2 / 20., 20., 6., 6. /
C
C Call the initialization routine.
C
CALL INIT3D (EYE, NU, NV, NW, ST1, LX, NY, IS2, IUNIT, S)
C
C Initialize the error indicator
C
IERROR = 1
C
C Create and plot the interlocking doughnuts.
C
JCENT1 = FLOAT(NV) *.5 - RBIG1 *.5
JCENT2 = FLOAT(NV) *.5 + RBIG2 *.5
DO 70 IBKWD = 1, NU
   I = NU + 1 - IBKWD
   FIMID = I - NU / 2
   DO 20 J = 1, NV
      FJMID1 = J - JCENT1
      FJMID2 = J - JCENT2
      DO 10 K = 1, NW
         FKMD = K - NW / 2
         F1 = SQRT (RBIG1 * RBIG1 / (FJMID1 * FJMID1 + FKMD * FKMD + .1))
         F2 = SQRT (RBIG2 * RBIG2 / (FIMID * FIMID + FJMID2 * FJMID2 + .1))
         FIP1 = (-F1) * FIMID
         FIP2 = (-F2) * FIMID
         FJP1 = (1. - F1) * FJMID1
         FJP2 = (1. - F2) * FJMID2
   10 CONTINUE
   DO 70 J = 1, NV
      FJMID1 = J - JCENT1
      FJMID2 = J - JCENT2
   20 CONTINUE
70 CONTINUE
FKP1 = (1.-F1)*FKMID
FKP2 = (1.-F2)*FKMID
TEMP = AMIN1(FIMID**2+FJP1**2+FKP1**2-RSML1**2,
FKMID**2+FIP2**2+FJP2**2-RSML2**2)

IF (TEMP .LE. 0.) IOBJS(J,K) = 1
IF (TEMP .GT. 0.) IOBJS(J,K) = 0

C Set proper words to 1 for drawing axes.
C
IF (I .NE. 1) GO TO 50
DO 30 K=1,NW
    IOBJS(1,K) = 1
30 CONTINUE
DO 40 J=1,NV
    IOBJS(J,1) = 1
40 CONTINUE
GO TO 60
50 CONTINUE
IOBJS(1,1) = 1
60 CONTINUE

C Call the draw and remember routine for this slab.
C
CALL DANDR (NV,NW,ST1,LX,NX,NY,IS2,IUNIT,S,IOBJS,MV)

C Select normalization transformation 0.
C
CALL GSELNT(O)

C Call WTSTR to write the plot title.
C
XC = CPUX(IX)
YC = CPUY(IY)
CALL WTSTR(XC,YC,'DEMONSTRATION PLOT FOR ISOSRFHR',2,0,0)
CALL GSELNT(ICN)

C Advance the frame.
C
CALL FRAME

IERROR = 0
WRITE (6,1001)
RETURN

1001 FORMAT (' ISOSRFHR TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR ISOSRFHR
PWRITX Example

SUBROUTINE TPWRTX (IERROR)

C
C PURPOSE
C
C To provide a demonstration of PWRITX
C and to test PWRITX with an example.

C USAGE
C
C CALL TPWRTX (IERROR)

C ARGUMENTS
C
C IERROR
C An integer variable
C = 0, if the test is successful,
C = 1, otherwise

C ON OUTPUT
C
C If the test is successful, the message
C PWRITX TEST EXECUTED--SEE PLOTS TO CERTIFY
C is written on unit 6.

C I/O
C
C In addition, four frames are produced. The
C first three frames contain complete
C character plots, and the fourth frame
C tests various settings of the function
C codes. To determine if the test is
C successful, it is necessary to examine these
C plots.

C PRECISION
C
C Single

C REQUIRED ROUTINES
C
C PWRITX

C REQUIRED GKS LEVEL
C
C OA

C LANGUAGE
C
C FORTRAN

C ALGORITHM
C
C TPWRTX calls the software character drawing
C subroutine PWRITX once for twelve different
C function codes for 46 separate characters.
C This produces a total of 552 characters on
C four separate plots. Each plot contains a
C grid of characters with the principle Roman
C characters in the first column and their
C other representations, produced with different
C function codes, across each row. Each function
C code has a mnemonic interpretation (e.g.,
C PRU - Principle Roman Upper, IGL - Indexical
C Greek Lower). In the first four plots, each
column is labeled with its function code.
The fifth plot invokes PWRITX with various function codes.

CHARACTER*1 DAT(48)

DAT contains the standard character set

DATA DAT(1), DAT(2), DAT(3), DAT(4), DAT(5), DAT(6), DAT(7), DAT(8),
1 DAT(9), DAT(10), DAT(11), DAT(12), DAT(13), DAT(14), DAT(15),
2 DAT(16), DAT(17), DAT(18), DAT(19), DAT(20), DAT(21), DAT(22),
3 DAT(23), DAT(24), DAT(25), DAT(26), DAT(27), DAT(28), DAT(29),
4 DAT(30), DAT(31), DAT(32), DAT(33), DAT(34), DAT(35), DAT(36),
5 DAT(37), DAT(38), DAT(39), DAT(40), DAT(41), DAT(42), DAT(43),
6 DAT(44), DAT(45), DAT(46), DAT(47), DAT(48) /
7 'A', 'B', 'C', 'D', 'E', 'F', 'G',
8 'H', 'I', 'J', 'K', 'L', 'M', 'N',
9 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
+ 'V', 'W', 'X', 'Y', 'Z', 'O', '1',
1 '2', '3', '4', '5', '6', '7', '8',
2 'g', '+', '-', '*', '(', ')', /
3 '$', '=' , ', ' , ', ', '/

Use normalization transformation 0

CALL GSELNT(0)

A separate frame is produced for each iteration through this loop

DO 160 K=1,4

Label the column and change the function code

DO 150 J=1,12
   XPOS = FLOAT(J*80-39) / 1024.
   GO TO ( 10, 20, 30, 40, 50, 60, 70, 80, 90, 100,
   1 110,120),J
10 CALL PWRITX (XPOS, .9375, 'PRU', 3, 16, 0, 0)
   CALL PWRITX (1./1024., 1./1024., ''PRU'', 5, 1, 0, 0)
   GO TO 130
20 CALL PWRITX (XPOS, .9375, 'PRL', 3, 16, 0, 0)
   CALL PWRITX (1./1024., 1./1024., ''PRL'', 5, 1, 0, 0)
   GO TO 130
30 CALL PWRITX (XPOS, .9375, 'IRU', 3, 16, 0, 0)
   CALL PWRITX (1./1024., 1./1024., ''IRU'', 5, 1, 0, 0)
   GO TO 130
40 CALL PWRITX (XPOS, .9375, 'IRL', 3, 16, 0, 0)
   CALL PWRITX (1./1024., 1./1024., ''IRL'', 5, 1, 0, 0)
   GO TO 130
50 CALL PWRITX (XPOS, .9375, 'KRU', 3, 16, 0, 0)
   CALL PWRITX (1./1024., 1./1024., ''KRU'', 5, 1, 0, 0)
   GO TO 130
CALL PWRITX (XPOS, .9375, 'KRL', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''KRL''', 5, 1, 0, 0)
GO TO 130

CALL PWRITX (XPOS, .9375, 'PGU', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''PGU''', 5, 1, 0, 0)
GO TO 130

CALL PWRITX (XPOS, .9375, 'PGL', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''PGL''', 5, 1, 0, 0)
GO TO 130

CALL PWRITX (XPOS, .9375, 'IGU', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''IGU''', 5, 1, 0, 0)
GO TO 130

CALL PWRITX (XPOS, .9375, 'IGL', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''IGL''', 5, 1, 0, 0)
GO TO 130

CALL PWRITX (XPOS, .9375, 'KGU', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''KGU''', 5, 1, 0, 0)
GO TO 130

CALL PWRITX (XPOS, .9375, 'KGL', 3, 16, 0, 0)
CALL PWRITX (1./1024., 1./1024., '''KGL''', 5, 1, 0, 0)

CONTINUE

DO 140 I=1,12
   YPOS = FLOAT( 980-I*80 ) / 1024.
   IF = I+(K-1)*12
   CALL PWRITX (XPOS, YPOS, DAT(IF), 1, 1, 0, 0)

CONTINUE

RETURN function to Principle Roman Upper to label column

CALL PWRITX (1./1024., 1./1024., '''PRU''', 5, 1, 0, 0)

CONTINUE

Label frame

CALL PWRITX (.5, 1000./1024.,
  + 'DEMONSTRATION PLOT FOR PWRITX', 29, 1, 0, 0)

CALL FRAME

CONTINUE
<table>
<thead>
<tr>
<th>Letter</th>
<th>Lowercase</th>
<th>Uppercase</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>d</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>e</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>f</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>g</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>h</td>
<td>H</td>
</tr>
<tr>
<td>I</td>
<td>i</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>j</td>
<td>J</td>
</tr>
<tr>
<td>K</td>
<td>k</td>
<td>K</td>
</tr>
<tr>
<td>L</td>
<td>l</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>m</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>n</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>o</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>p</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>q</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>r</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>s</td>
<td>S</td>
</tr>
<tr>
<td>T</td>
<td>t</td>
<td>T</td>
</tr>
<tr>
<td>U</td>
<td>u</td>
<td>U</td>
</tr>
<tr>
<td>V</td>
<td>v</td>
<td>V</td>
</tr>
<tr>
<td>W</td>
<td>w</td>
<td>W</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>y</td>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
<td>z</td>
<td>Z</td>
</tr>
</tbody>
</table>

DEMONSTRATION PLOT FOR PWRITX

```
PRU  PRL  IRU  IRL  KRU  KRL  PGU  PGL  IGU  IGL  KGU  KGL
A    a    A    a    A    a    @    A    @    A    A    A    A
B    b    B    b    B    b    $    B    $    B    B    B    B
C    c    C    c    C    c    t    C    t    C    C    C    C
d    d    d    d    d    t    d    t    d    d    d    d    t
e    e    e    e    e    e    e    e    e    e    e    e    e
f    f    f    f    f    f    f    f    f    f    f    f    f
G    g    g    g    g    g    g    H    H    H    H    H    H
h    h    h    h    h    s    @    s    @    s    @    s    @
i    i    i    i    i    i    i    i    i    i    i    i    i
j    j    j    j    j    j    k    k    k    k    k    k    k
K    k    k    k    k    k    k    k    k    k    k    k    k
L    l    L    l    L    l    L    L    L    L    L    L    L
M    m    M    m    M    m    M    m    M    M    M    M    M
```
<table>
<thead>
<tr>
<th>DEMONSTRATION PLOT FOR PWRITX</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRU</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>Y y</td>
</tr>
<tr>
<td>Z z</td>
</tr>
<tr>
<td>0 0</td>
</tr>
<tr>
<td>1 1</td>
</tr>
<tr>
<td>2 2</td>
</tr>
<tr>
<td>3 3</td>
</tr>
<tr>
<td>4 4</td>
</tr>
<tr>
<td>5 5</td>
</tr>
<tr>
<td>6 6</td>
</tr>
<tr>
<td>7 7</td>
</tr>
<tr>
<td>8 8</td>
</tr>
<tr>
<td>9 9</td>
</tr>
</tbody>
</table>

Partial alphanumeric characters appear in inconsistent formats and are not coherent.
Tests:

Upper and lower case function codes
Sub- and Super-scripting and Normal function codes
Down and Across function codes
Direct Character access
X and Y coordinate control, Carriage control function codes
Orientation of string (argument to PWRITX)

Test direct character access and string orientation

CALL PWRITX(.5,.5,'''546''',5,3,0,0)
CALL PWRITX(.5,.5,'''H9L''ANGLE OF ''U''30',19,18,30,-1)
CALL PWRITX(.5,.5,'''H-9L''ANGLE OF ''U''190',21,18,190,-1)

Upper and Lower case

CALL PWRITX(.65,.25,'''L2''LOWER, 3 ''U3''UPPER',24,1,0,0)

Level definitions (sub and superscripting)

CALL PWRITX(.95,.15,'''NU''NORMAL'.
-31,0,0,1)
CALL PWRITX(.95,.05,'''N''SHOW ''U1''U''S''SE OF''NU''NORMAL'.
-30,0,0,1)

Direction definitions

CALL PWRITX(.05,.5,'''DO''D''WNA''A''CROSS',16,0,0,-1)

Coordinate definitions

CALL PWRITX(.3,.85,
-''L''U''V7''S''V7''E''V7'' ''V7U1''V''V7'' FOR VERTICAL STEPS',
-49,0,0,0)
CALL PWRITX(.25,.6,
-''U''SHIFT''H11''.'''H11''.''H11''.''H11''.RIGHT''.
-37,14,90,-1)
CALL PWRITX(.45,.6,
-'SHIFT''H-30''.''H-11''.''H-11''.''H-11''.LEFT'.
-37,14,90,-1)
CALL PWRITX(.8,.8,'''L3''USE C''CL''FOR''C''CARRIAGE''C''CARRIAGE''C''CARRIAGE'',
-37,16,0,0)
CALL PWRITX(.1,.1,'''UX50Y50''( X50, Y50 )''X99Y99''( X99, Y99 ')
-41,14,0,-1)

Label frame

CALL PWRITX(.5,1000./1024.,
CALL FRAME

C

IERROR = 0
WRITE (6,1001)
RETURN

C

1001 FORMAT (' PWRITX TEST EXECUTED--SEE PLOTS TO CERTIFY')

C

END

use \( v \) for vertical steps

use \( c \) for carriage returns

DO 

W

N

ACROSS

2 lower, 3 upper

\(( X99, Y99 \) )

\(( X50, Y50 \) )

SHOW use of normal

August 1987
PWRITY Example

SUBROUTINE TPWRY (IERROR)

PURPOSE
To provide a simple demonstration of entry PWRITY.

USAGE
CALL TPWRY (IERROR)

ARGUMENTS

ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise

I/O
If the test is successful, the message
PWRITY TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 1 frame is produced on the machine graphics device. In order to determine if the test was successful, it is necessary to examine the plot.

PRECISION
Single

LANGUAGE
FORTRAN 77

REQUIRED ROUTINES
PWRITY

REQUIRED GKS LEVEL
OA

Initialize the error parameter.
IERROR = 0

Define normalization transformation 1 and log scaling.
CALL SET(0.0, 1.0, 0.0, 1.0, 1.0, 1024.0, 1.0, 1024.0, 1)

Label the frame.
CALL PWRITY(512.0, 950.0,
 1 'DEMONSTRATION PLOT FOR PWRITY',
 2 29,2,0,0)

Test PWRITY for different size characters.
CALL PWRITY (10.0, 900.0,'SIZE TEST',9,0,0,-1)
CALL PWRITY (10.0,850.0,'SIZE TEST',9,1,0,-1)

August 1987
CALL PWRITY (10.0, 775.0, 'SIZE TEST', 9, 2, 0, -1)
CALL PWRITY (10.0, 675.0, 'SIZE TEST', 9, 3, 0, -1)
CALL PWRITY (10.0, 525.0, 'SIZE TEST', 9, 4, 0, -1)
CALL PWRITY (10.0, 375.0, 'SIZE TEST', 9, 5, 0, -1)

C
C Test PWRITY for different character orientations.
C
CALL PWRITY (600.0, 600.0, 'THETA TEST', 10, 2, 0*90, -1)
CALL PWRITY (600.0, 600.0, 'THETA TEST', 10, 2, 1*90, -1)
CALL PWRITY (600.0, 600.0, 'THETA TEST', 10, 2, 2*90, -1)
CALL PWRITY (600.0, 600.0, 'THETA TEST', 10, 2, 3*90, -1)

C
C Test various centering options.
C
CALL PWRITY (512.0, 160.0, 'CENTR TEST', 10, 2, 0, 0)
CALL PWRITY (512.0, 85.0, 'CENTR TEST', 10, 2, 0, -1)
CALL PWRITY (512.0, 235.0, 'CENTR TEST', 10, 2, 0, 1)
CALL FRAME
C
WRITE (6, 1001)
RETURN
C
1001 FORMAT (' PWRITY TEST EXECUTED--SEE PLOTS TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR PWRITY

SIZE TEST
SIZE TEST
SIZE TEST

CENTR TEST
CENTR TEST
CENTR TEST

August 1987 537
SUBROUTINE TPWRZI (IERROR)

To provide a simple demonstration of entry PWRZI with the ISOSRF utility.

CALL TPWRZI (IERROR)

IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise

If the test is successful, the message
PWRZI TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 1
frame is produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plot.

Single

PWRZI, ISOSRF

OA

FORTRAN 77

A function of 3 variables is defined and the
values of the function on a 3-D rectangular
grid are stored in an array. This test routine
then calls ISOSRF to draw an iso-surfaced
plot of the function. PWRZI is then called 3
times to label the front, side, and back of
the picture.

T(21,31,19),SLAB(33,33),EYE(3)

Define the center of a plot title string on a square grid of size
0. to 1.

DATA TX/0.4375/, TY/0.9667/

DATA NU,NV,NW/21,31,19/
DATA RBIG1,RBIG2,RSML1,RSML2/6.,6.,2.,2./
DATA TISO/O./
DATA MUVWP2/33/
DATA IFLAG/-7/
C
C Initialize the error parameter.
C
IERROR = 1
C
C Fill the 3-D array to be plotted.
C
JCENT1 = FLOAT(NV)*.5-RBIG1*.5
JCENT2 = FLOAT(NV)*.5+RBIG2*.5
DO 30 I=1,NU
   FIMID = I-NU/2
   DO 20 J=1,NV
      FJMID1 = J-JCENT1
      FJMID2 = J-JCENT2
      DO 10 K=1,NW
         FKMID = K-NW/2
         F1 = SQRT(RBIG1*RBIG1/(FJMID1*FJMID1+FKMID*FKMID+.1))
         F2 = SQRT(RBIG2*RBIG2/(FIMID*FIMID+FJMID2*FJMID2+.1))
         FIP1 = (1.-F1)*FIMID
         FIP2 = (1.-F2)*FIMID
         FJP1 = (1.-F1)*FJMID1
         FJP2 = (1.-F2)*FJMID2
         FKP1 = (1.-F1)*FKMID
         FKP2 = (1.-F2)*FKMID
         T(I,J,K) = AMIN1(FIMID*FIMID+FJP1*FJP1+FKP1*FKP1-
                        RSML1*RSML1,
                        FKMID*FKMID+FIP2*FIP2+FJP2*FJMID2-RSML2*RSML2)
   10 CONTINUE
   20 CONTINUE
30 CONTINUE
C
C Define the eye position.
C
EYE(1) = 100.
EYE(2) = 150.
EYE(3) = 125.
C
C Select normalization transformation number 0.
C
CALL GSELNT (0)
C
C Label the plot.
C
CALL WTSTR (TX,TY,'DEMONSTRATION PLOT FOR PWRZI',2,0,0)
C
C Test ISOSRF with subarray T.
C
MU = NU/2
MV = NV/2
MW = NW/2
MUVWP2 = MAXO(MU,MV,MW)+2
CALL ISOSRF (T(MU,MV,MW), NU,MU,NV,MV,MW,EYE,MUVWP2,SLAB,TISO,
1      IFLAG)
ISIZE = 35
CALL PWRZI (5.,16.,.5,'FRONT',5,ISIZE,-1,3,0)
CALL PWRZI (11.,7.5,.5,'SIDE',4,ISIZE,2,-1,0)
CALL PWRZI (5.,1.,5.,' BACK BACK BACK BACK BACK',25,ISIZE,-1,3,0)
CALL FRAME
IERRO = 0
C
WRITE (6,1001)
RETURN
C
1001 FORMAT (' PWRZI TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END

DEMONSTRATION PLOT FOR PWRZI
BACK BACK BACK BACK
SIDE
FRONT

August 1987
541
PWRZS + SRFACE Example

SUBROUTINE TPWRZS (IERROR)
C
C PURPOSE
C To provide a simple demonstration of entry PWRZS with the SRFACE utility.
C
C USAGE
C CALL TPWRZS (IERROR)
C
C ARGUMENTS
C IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
I/O
If the test is successful, the message
PWRZS TEST EXECUTED--SEE PLOTS TO CERTIFY
C
is printed on unit 6. In addition, 1 frame is produced on the machine graphics
device. In order to determine if the test was successful, it is necessary to examine
the plot.

C PRECISION
Single
C
C REQUIRED ROUTINES
PWRZS, SRFACE
C
C REQUIRED GKS LEVEL
OA
C
C LANGUAGE
FORTRAN 77
C
C ALGORITHM
A function of 2 variables is defined and the
values of the function on a 2-D rectangular
grid are stored in an array. This routine
calls SRFACE to draw a surface representation
of the array values. PWRZS is then called 3
times to label the front, side, and back of
the picture.

DIMENSION Z(20,30) ,X(20) ,Y(30) ,MM(20,30,2),
1 S(6)
C
Load the SRFACE common block needed to supress a NEWFM call.
C
COMMON /SRFIP1/ IFR ,ISTP ,IROT ,IDRX,
1 IDR ,IDRZ ,IUPPER ,ISKIRT ,
2 NCLA ,THETA ,HSKIRT ,CHI,
3 CLO ,CINC ,ISPVAL
C Define the center of a plot title string on a square grid of size 0. to 1.
C
DATA TX/0.4375/, TY/0.9667/
C
C Specify grid loop indices and a line of sight.
C
DATA M/20/, N/30/
DATA S/4.,5.,3.,0.,0.,0./
C
C Initial the error parameter.
C
IERRO = 1
C
C Define the function values and store them in the Z array.
C
DO 10 I=1,M
   X(I) = -1.+FLOAT(I-1)/FLOAT(M-1)*2.
10 CONTINUE
DO 20 J=1,N
   Y(J) = -1.+FLOAT(J-1)/FLOAT(N-1)*2.
20 CONTINUE
DO 40 J=1,N
   DO 30 I=1,M
      Z(I,J) = EXP(-2.*SQRT(X(I)**2+Y(J)**2))
   30 CONTINUE
40 CONTINUE
C
C Set SREACE parameters to suppress the FRAME call and draw contours.
C
IFR = 0
IDRZ = 1
C
C Select normalization trans number 0.
C
CALL GSELNT (0)
C
C Label the plot.
C
CALL WTSTR (TX,TY,'DEMONSTRATION PLOT FOR PWRZS',2,0,0)
C
C Draw the surface plot.
C
CALL SRFACE (X,Y,Z,MM,M,M,N,S,O.)
C
C Put the PWRZS labels on the picture.
C
ISIZE = 35
CALL PWRZS (0.,1.1,0.,'FRONT',5,ISIZE,-1,3,0)
CALL PWRZS (1.1,0.,0.,'SIDE',4,ISIZE,2,-1,0)
CALL PWRZS (0.,-1.1,2,' BACK BACK BACK BACK',25,ISIZE,-1,1,3,0)
CALL FRAME
C
  IERROR = 0
  WRITE (6,1001)
C
C Restore the SRFACE parameters to their default values.
C
IFR = 1
IDRZ = 0
C
RETURN
C
1001 FORMAT (' PWRZS TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END

DEMONSTRATION PLOT FOR PWRZS
PWRZT + THREED Example

SUBROUTINE TPWRZT (IERROR)
C
C PURPOSE
C
C
C USAGE
C
C ARGUMENTS
C
To provide a simple demonstration of entry PWRZT with the THREED utility.
CALL TPWRZT (IERROR)
C
C ON OUTPUT
C
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
I/O
If the test is successful, the message
PWRZT TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 1 frame is produced on the machine graphics
device. In order to determine if the test was successful, it is necessary to examine
the plot.
C
C PRECISION
Single
C
C REQUIRED ROUTINES
PWRZT, THREED
C
C REQUIRED GKS LEVEL
OA
C
C LANGUAGE
FORTRAN 77
C
C ALGORITHM
The THREED package is called to establish a
3-D projection onto 2 space, and to draw the
axis lines. PWRZT is then called to label
the axes for a 3 space plot.
C
C EYE contains the (U,V,Z) coordinate of the eye position.
REAL EYE(3)
DATA EYE(1), EYE(2), EYE(3) /3.5, 3.0, 5.0/
C
C Initialize the error parameter.
C
IERROR = 1
C
C Select normalization transformation number 0.
C
CALL GSELNT (0)
C
August 1987
547
C A call to SET3 establishes the mapping of 3 space coordinates onto
C the coordinate system of the graphics device.
C
CALL SET3 (.1,.9,.1,.9,0.,1.,0.,1.,0.,1.,EYE)
C
C Draw the 3 space axes.
C
CALL LINE3 (0.,0.,0.,0.,0.,1.)
CALL LINE3 (0.,0.,0.,0.,1.,0.)
CALL LINE3 (0.,0.,0.,1.,0.,0.)
C
C PWRZT is used to label each of the axes and the plot
C
ICNT = 0
ISIZE = 30
LINE = 2
ITOP = 3
CALL PWRZT (.5,.1,'V-AXIS',6,ISIZE,LINE,ITOP,ICNT)
C
LINE = -1
ITOP = 3
CALL PWRZT (.5,0.,.1,'U-AXIS',6,ISIZE,LINE,ITOP,ICNT)
C
LINE = 3
ITOP = -2
CALL PWRZT (0.,.1,.5,'Z-AXIS',6,ISIZE,LINE,ITOP,ICNT)
C
LINE = 2
ITOP = -1
ISIZE = 30
ICNT = -1
CALL PWRZT (.5,.2,0.,'DEMONSTRATION OF PWRZT WITH THREED',
1                      34,ISIZE,LINE,ITOP,ICNT)
CALL FRAME
C
IERROR = 0
WRITE (6,1001)
C
RETURN
1001 FORMAT (' PWRZT TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
SRFACE Example

SUBROUTINE TSRFAC (IERROR)
C
C PURPOSE
C
To provide a simple demonstration of SRFACE.
C
C USAGE
C
CALL TSRFAC (IERROR)
C
C ARGUMENTS
C
IERROR
An integer variable

= 0, if the test was successful,

= 1, the test was not successful.
C
C I/O
C
SRFACE TEST EXECUTED--SEE PLOT TO CERTIFY
C
is printed on unit 6. In addition, 2
frames are produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plots.
C
C PRECISION
C
Single
C
C LANGUAGE
C
FORTRAN 77
C
C REQUIRED ROUTINES
C
SRFACE
C
C REQUIRED GKS LEVEL
C
OA
C
C ALGORITHM
C
The function
C
Z(X,Y) = .25*(X + Y + 1./((X-.1)**2+Y**2+.09)
-1./((X+.1)**2+Y**2+.09)
C
for X = -1. to +1. in increments of .1, and
Y = -1.2 to +1.2 in increments of .1,
C
is computed. Then, entries EZSRFC and SURFACE
C
are called to generate surface plots of Z.
C
C HISTORY
C
SURFACE was first written in April 1979 and
C
converted to FORTRAN 77 and GKS in March 1984.
C
XX contains the X-direction coordinate values for Z(X,Y); YY contains
C the Y-direction coordinate values for Z(X,Y); Z contains the function
C values; S contains values for the line of sight for entry SRFACE;
C WORK is a work array; ANGH contains the angle in degrees in the X-Y
C plane to the line of sight; and ANGV contains the angle in degrees
C from the X-Y plane to the line of sight.
REAL XX(21), YY(25), Z(21, 25), S(6)
1 WORK(1096)

DATA S(1), S(2), S(3), S(4), S(5), S(6)/
1 -8.0, -6.0, 3.0, 0.0, 0.0, 0.0/

DATA ANGH/45./, ANGV/15./

C Specify coordinates for plot titles. The values CX and CY C define the center of the title string in a 0. to 1. range.
DATA CX/.5/, CY/.9/

C Initialize the error parameter.
IERROR = 0

C Fill the XX and YY coordinate arrays as well as the Z function array.
DO 20 I=1, 21
   X = .1*FLOAT(I-11)
   XX(I) = X
   DO 10 J=1, 25
      Y = .1*FLOAT(J-13)
      YY(J) = Y
      Z(I,J) = (X+Y+1./((X-.1)**2+Y**2+.09)
               - 1./((X+.1)**2+Y**2+.09))*.25
   10 CONTINUE
20 CONTINUE

C Select the normalization transformation 0.
CALL GSELNT(0)

C Frame 1 -- The EZSRFC entry.

C Add the plot title using GKS calls.
CALL GSTXAL(2, 3)

C Set the text alignment to center the string in horizontal and vertical
CALL GSTXAL(2, 3)

C Set the character height.
CALL GSCHH(.016)

C Write the text.
CALL GTX(CX,CY,'DEMONSTRATION PLOT FOR EZSRFC ENTRY OF SRFACE')
CALL EZSRFC(Z, 21, 25, ANGH, ANGV, WORK)
DEMONSTRATION PLOT FOR EZSRFC ENTRY OF SRFACE

August 1987
C C Frame 2 -- The SRFACE entry.
C
C Add the plot title.
C
C Set the text alignment to center the string in horizontal and vertical
C
CALL GSTXAL(2,3)
C
C Set the character height.
C
CALL GSCHH(.016)
C
C Write the text.
C
CALL GTX(CX,CY,'DEMONSTRATION PLOT FOR SRFACE ENTRY OF SRFACE')
C
CALL SRFACE (XX,YY,Z,WORK,21,21,25,S,O.)
C
C This routine automatically generates frame advances.
C
WRITE (6,1001)
C
RETURN
C
1001 FORMAT (' SRFACE TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR SRFACE ENTRY OF SRFACE
STRMLN Example

SUBROUTINE TSTRML (IERROR)
C
C PURPOSE To provide a simple demonstration of STRMLN.
C
C USAGE CALL TSTRML (IERROR)
C
C ARGUMENTS
C
C ON OUTPUT
C
C IERROR An integer variable
C = 0, if the test was successful,
C = 1, the test was not successful.
C
C I/O If the test is successful, the message
C
C STRMLN TEST EXECUTED--SEE PLOT TO CERTIFY
C
C is printed on unit 6. In addition, 1
C frame is produced on the machine graphics
C device. In order to determine if the test
C was successful, it is necessary to examine
C the plot.
C
C PRECISION Single
C
C LANGUAGE FORTRAN 77
C
C REQUIRED ROUTINES STRMLN
C
C REQUIRED GKS LEVEL OA
C
C ALGORITHM Routine TSTRML calls routine STRMLN to
C produce a plot which depicts the flow and
C magnitude of a vector field.
C
REAL U(21,25) ,V(21,25) ,WRK(1050)
C
Specify coordinates for plot titles. The values TX and TY
C define the center of the title string in a 0. to 1. range.
C
DATA TX/.5/ ,TY/ .9765/
C
Set the grid dimensions.
C
DATA NH,NV/21,25/
C
Initialize the error parameter.
C
IERROR = 1
C
August 1987
557
C Specify horizontal and vertical vector components U and V on
C the rectangular grid.
C
TPIMX = 2.*3.14/FLOAT(NH)
TPJMX = 2.*3.14/FLOAT(NV)
DO 20 J=1,NV
   DO 10 I=1,NH
      U(I,J) = SIN(TPIMX*(FLOAT(I)-1.))
      V(I,J) = SIN(TPJMX*(FLOAT(J)-1.))
   10 CONTINUE
20 CONTINUE
C
C Select normalization transformation 0.
C
CALL GSELNT (0)
C
C Call WTSTR to write the plot title.
C
CALL WTSTR (TX,TY, 'DEMONSTRATION PLOT FOR ROUTINE STRMLN',2,1,0,0)
C
C Define normalization transformation 1, and set up log scaling.
C
CALL SET(0.1, 0.9, 0.1, 0.9, 1.0, 21., 1.0, 25.,1)
C
C Draw the plot perimeter.
C
CALL PERIM(1.0,1.0)
C
C Call STRMLN for vector field streamlines plot.
C
CALL STRMLN (U,V,WRK,NH,NH,NV,1,IER)
C
CALL FRAME
C
IERERROR = 0
WRITE (6,1001)
RETURN
C
1001 FORMAT (' STRMLN TEST EXECUTED--SEE PLOT TO CERTIFY')
C
END
DEMONSTRATION PLOT FOR ROUTINE STRMLN
THREED Example

SUBROUTINE TTHREE (IERROR)

C PURPOSE
To provide a simple demonstration of THREED.

C USAGE
CALL TTHREE (IERROR)

C ARGUMENTS
C
C ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
= 1, the test was not successful.

C I/O
If the test is successful, the message

THREED TEST EXECUTED--SEE PLOT TO CERTIFY

is printed on unit 6. In addition, 1
frame is produced on the machine graphics
device. In order to determine if the test
was successful, it is necessary to examine
the plot.

C PRECISION
Single

C LANGUAGE
FORTRAN 77

C REQUIRED ROUTINES
THREED

C REQUIRED GKS LEVEL
OA

C ALGORITHM
Routine TTHREE calls SET3 to establish a
mapping between the plotter addresses and
the user's volume, and to indicate the
coordinates of the eye position from which
the lines to be drawn are viewed. Next,
the volume perimeters and associated tick
marks are drawn by calls to PERIM3. The
selected latitude and longitude lines of
a sphere are then drawn.

C HISTORY
THREED was originally written in November
1976 and converted to FORTRAN 77 and GKS
in July 1984.

REAL EYE(3),X(31),Y(31),Z(31)

C Specify the arguments to be used by routine SET3 on a plot
C grid in the address range of 0. to 1. In each coordinate direction,
C the values RXA, RXB, RYA, and RYB define the portion of the address
C space to be used in making the plot. UC, UD, VC, VD, WC, and WD
C define a volume in user coordinates which is to be mapped onto the
C portion of the viewing surface as specified by RXA, RXB, RYA, and RYB.

```
DATA RXA/0.097656/, RXB/0.90236/, RYA/0.097656/, RYB/0.90236/
DATA UC/-1./, UD/1./, VC/-1./, VD/1./, WC/-1./, WD/1./
DATA EYE(1),EYE(2),EYE(3)/10.,6.,3./
DATA TX/0.4374/, TY/0.9570/
DATA PI/3.1415926535898/
```

C Select normalization transformation 0.
C
```
CALL GSELNT (0)
```

C Call SET3 to establish a mapping between the plotter addresses
C and the user's volume, and to indicate the coordinates of the
C eye position from which the lines to be drawn are viewed.
C
```
CALL SET3(RXA,RXB, RYA, RYB, UC, UD, VC, VD, WC, WD, EYE)
```

C Call PERIM3 to draw perimeter lines and tick marks.
C
```
CALL PERIM3(2,5,1,10,1,-1.)
CALL PERIM3(4,2,1,1,2,-1.)
CALL PERIM3(2,10,4,5,3,-1.)
```

C Define and draw latitudinal lines on the sphere of radius one
C having its center at (0.,0.,0.)
C
```
DO 10 J=1,18
   THETA = FLOAT(J)*PI/9.
   CT = COS(THETA)
   ST = SIN(THETA)
   DO 20 K=1,31
      PHI = FLOAT(K-16)*PI/30.
      Z(K) = SIN(PHI)
      CP = COS(PHI)
      X(K) = CT*CP
      Y(K) = ST*CP
   20 CONTINUE
   CALL CURVE3(X,Y,Z,31)
10 CONTINUE
```

C Define and draw longitudinal lines on the sphere of radius one
C having its center at (0.,0.,0.)
C
```
DO 30 K=1,5
   PHI = FLOAT(K-3)*PI/6.
   SP = SIN(PHI)
   CP = COS(PHI)
   DO 40 J=1,31
      TUETA = FLOAT(J-1)*PI/15.
      X(J) = COS(TUETA)*CP
```

August 1987
562
THREED

\[ Y(J) = \sin(TUETA) \cdot CP \]
\[ Z(J) = SP \]

40 CONTINUE
   CALL CURVE3(X, Y, Z, 31)
30 CONTINUE

C
C Add a plot title.
C
   CALL WTSTR(TX, TY, 'DEMONSTRATION PLOT FOR ROUTINE THREED', 2, 0, 0)
C
C Advance the frame.
C
   CALL FRAME
C
   IERROR = 0
   WRITE(6, 1001)
   RETURN
C
1001 FORMAT (' THREED TEST EXECUTED--SEE PLOT TO CERTIFY')
END
VELVCT Example

SUBROUTINE TVELVC (IERROR)
C
C PURPOSE
C
C To provide a simple demonstration of VELVCT.
C
C USAGE
C
C CALL TVELVC (IERROR)
C
C ARGUMENTS
C
C
C
C
C
C
C ON OUTPUT
C
C
C
C
C
C
C I/O
C
C
C
C
C
C
C PRECISION
C
C Single
C
C LANGUAGE
C
C FORTRAN 77
C
C REQUIRED ROUTINES
C
C VELVCT
C
C REQUIRED GKS LEVEL
C
C OA
C
C ALGORITHM
C
C This test program calls entries EZVEC and VELVCT. Each call produces a plot of a
C vector field obtained from the function
C
C  \[
C  Z(X,Y) = X + Y + 1./((X-.1)**2+Y**2+.09)
C  \]
C
C  \[
C  -1./((X+.1)**2+Y**2+.09),
C  \]
C
C by using the direction of the Z gradient vectors and the logarithm of the absolute
C value of the components.
C
C HISTORY
C
C Originally written in November 1976.
C
C Converted to FORTRAN 77 and GKS in July 1984.
C
C DIMENSION
C
C U(21,25) , V(21,25)
C
C Specify coordinates for a plot title.
C
August 1987

565
DATA IX/94/, IY/1000/
C
C Specify VELVCT arguments.
C
DATA FLO/O./, HI/O./, NSET/O/, LENGTH/O/, ISPV/O/, SPV/O./
C
C Initialize the error parameter.
C
IERROR = 1
C
C Specify velocity field functions U and V.
C
M = 21
N = 25
DO 20 I=1,M
   X = .1*FLOAT(I-11)
   DO 10 J=1,N
      Y = .1*FLOAT(J-13)
      DZDX = 1. - 2.*((X-.10)/(X-.10)**2+Y**2+.09)**2
           + 1.2*((X+.10)/(X+.10)**2+Y**2+.09)**2
      DZDY = 1. - 2.*Y/((X-.10)**2+Y**2+.09)**2
           + 1.2*Y/((X+.10)**2+Y**2+.09)**2
      UVMAG = ALOG(SQRT(DZDX*DZDX+DZDY*DZDY))
      UVDIR = ATAN2(DZDY,DZDX)
      U(I,J) = UVMAG*COS(UVDIR)
      V(I,J) = UVMAG*SIN(UVDIR)
10  CONTINUE
20  CONTINUE
C
CALL GQCNTN (IERR, ICN)
C
C Select normalization transformation 0.
C
CALL GSELNT(0)
C
C Call WTSTR to write the plot title.
C
   X = CPUX(IX)
   Y = CPUY(IY)
   CALL WTSTR (X, Y, 'DEMONSTRATION PLOT FOR ENTRY EZVEC OF VELVCT',
              1 2, 0, -1)
   CALL GSELNT(ICN)
C
C Call EZVEC for a default velocity field plot.
C
   CALL EZVEC (U, V, M, N)
DEMONSTRATION PLOT FOR ENTRY EZVEC OF VELVCT

MAXIMUM VECTOR

0.371E+01

August 1987
CALL VELVCT to generate the user tuned velocity field plot.

CALL VELVCT (U,M,V,M,N,FLO,HI,NSET,LENGTH,ISPV,SPV)
CALL GQCNTN(IERR,ICN)

CALL GSELNT(0)

CALL WTSTR to write the plot title.

CALL GSELNT(ICN)
CALL FRAME

IERROR = 0
WRITE (6,1001)
RETURN

1001 FORMAT (' VELVCT TEST EXECUTED--SEE PLOTS TO CERTIFY')

END
DEMONSTRATION PLOT FOR ENTRY VELVCT OF VELVCT

August 1987

569
Complex Uses of Higher-level Utilities
Complex Uses of Higher-level Utilities

This section contains six examples of more sophisticated uses of the higher-level graphics utilities. A complete program with the plots produced is provided. The examples are as follows:

- Three curves on the same plot, using AUTOGRAPH with a user-defined dash pattern.

- A CONREC plot using a polar coordinate transformation.

- A CONREC plot overlaid on an EZMAP plot.

- Another CONREC plot overlaid on EZMAP.

- A CONREC plot with several user-defined parameters and a modified background using GRIDAL.

- A VELVCT plot overlaid on an EZMAP plot.

Since several of the examples involve superimposing the plot from one utility onto the plot of another, a brief discussion of how to transform the coordinate output from certain utilities follows.

Transforming Coordinate Output

Several of the utilities contain the facility to transform the lines drawn from default positions to other positions on the screen, resulting in a new shape for the picture. The utilities implemented with this option are:

- the CONREC family.

- the CONRAN family.

- ISOSRF.

- STRMLN.

- VELVCT.

The way the transformation option is implemented is to include arithmetic statement functions in the selected utilities that provide the coordinate transformations. By default, these statement functions are set to the identity transformation. If a non-identity transformation is desired, the statement functions are replaced appropriately. (If the transformation is a complicated one, it will be desirable to delete the statement functions and add external routines with the same names to do the transforming.) The arithmetic statement functions for transformations are names FX (for converting X coordinates) and FY (for converting Y coordinates). Initially, the statement functions FX and FY are
NCAR Graphics User's Guide

\[
\begin{align*}
FX (X) &= X \\
FY (Y) &= Y
\end{align*}
\]

These are the functions which must be replaced in the non-default case.

The range of the arguments to the statement functions depends on the routine in use. The examples supplied here illustrate this. The coordinate spaces for CONREC and VELVCT are FLOAT(1) to FLOAT(N) by FLOAT(1) to FLOAT(M) where the input array is dimensioned for N x M.

Here is a list of file names and the subroutines in the given file that contain references to arithmetic statement functions FX and FY.

<table>
<thead>
<tr>
<th>File</th>
<th>Subroutines Containing FX and FY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONRCQCK</td>
<td>MAXMIN</td>
</tr>
<tr>
<td></td>
<td>QUICK</td>
</tr>
<tr>
<td>CONREC</td>
<td>MINMAX</td>
</tr>
<tr>
<td></td>
<td>DRLINE</td>
</tr>
<tr>
<td>CONRCSPR</td>
<td>MINMAX</td>
</tr>
<tr>
<td></td>
<td>DRLINE</td>
</tr>
<tr>
<td>CONCOM</td>
<td>CONGEN</td>
</tr>
<tr>
<td>CONTERP</td>
<td>CONTLK</td>
</tr>
<tr>
<td>CONRAQ</td>
<td>CONTOR</td>
</tr>
<tr>
<td>CONRAN</td>
<td>CONPMM</td>
</tr>
<tr>
<td>CONRAS</td>
<td>CONPMS</td>
</tr>
<tr>
<td>ISOSRF</td>
<td>DCNTR</td>
</tr>
<tr>
<td>STRMLN</td>
<td>DRWSTR</td>
</tr>
<tr>
<td>VELVCT</td>
<td>VELVCT</td>
</tr>
</tbody>
</table>

Also, there are some special considerations pertaining to superimposing VELVCT plots onto other plots, with regard to the length of the vectors. See the VELVCT documentation in the "Utilities" section.
The following FORTRAN code and interspersed plots show some more detailed examples of the package. Six plot examples are given. This code is a complete program that runs on one of the Scientific Computing Division's UNIX™ computers. Only two changes need to be made in the code to convert this program to run on any computer; these changes are indicated in the comments.

```fortran
PROGRAM COMPLEX
  C  This program produces plots illustrating non-trivial usage of the NCAR Utilities. This program is an exact copy of an implementation that runs on a UNIX machine at NCAR. To implement this program on another machine only the following steps should be required:
  C
  1.) Modify the OPEN to the EZMAP dataset that appears in the main program.
  C
  2.) Modify the OPEN to the file of random numbers RANFDAT in subroutine RAND.

  The utilities AUTOGRAPH, VELVCT, CONREC, EZMAP, GRIDAL, and DASHCHAR are required. Since CONREC and VELVCT plots are overlaid on EZMAP, the arithmetic statement functions for FX and FY in these two utilities should be changed to the statement:

  EXTERNAL FX,FY

  This will force the functions FX and FY supplied in this package to be loaded. The documentation for FX explains the transformations provided.

  PARAMETER ( M=70 , N=150 , NPR=155)
  PARAMETER (PI=3.14159 , TWOPI=2.*PI , EPS=PI/6.)
  DIMENSION A(M,NPR),B(M,N)
  EQUIVALENCE (A(1,5),B(1,1))
  COMMON /TRANS/ MA,NA,ITRANS,A1,A2,D1,D2,ALNMN,ALNMX,ALTMN,ALTMX
  CHARACTER*1 ZERO
  CHARACTER*16 LDASH(1)
  DATA LDASH(1) /'$$'$$''$$''$$''$$''$$'/'

  OPEN (UNIT=1,FILE='/u2/gp/gkslib/demos/ezmap.out',
          STATUS='OLD',FORM='UNFORMATTED')
  C
  Open GKS, open workstation, activate workstation.
  CALL GOPKS (6,ISZ)
  CALL GOPWK (1, 2, 1)
  CALL GACWK (1)
  ZERO = CHAR(O)

TMUNIX is a trademark of Bell Laboratories.
```

August 1987 575
C
C Generate the input array.
C
CALL GENARA(B,A,M,N)
C
C Frame 1: AUTOGRAPH
C
CALL AGSETI('DASH/LENGTH.',16)
CALL DISPLA(2,0,1)
DO 160 J=1,3
   YT = 0.3*FLOAT(J)
   YB = 0.3*FLOAT(J-1)+.05
   CALL SET(.1,.9,YB,YT,0.,1.,0.,1.,1)
   CALL ANOTAT(ZERO,ZERO,3,2,1,LDASH)
   CALL EZY(A(1,3*J),60,'$
160 CONTINUE
CALL FRAME
Frame 2: CONREC + FX + FY (Polar Projection)

ITRANS = 1
D1 = 20.
MA = 30
NA = 70
CALL GENARA(B,A,MA,NA+12)
XMIN = -60.
XMAX = 60.
YMIN = -60.
YMAX = 80.
CALL SET(.1,.9,.1,.9,XMIN,XMAX,YMIN,YMAX,1)
CALL LABMOD ('(F5.0)','(F5.0)',5,5,0,0,0,0,0,0)
CALL PERIML(6,2,6,2)
D2 = 30.
A1 = EPS
A2 = TWOPI-.5*A1
DELY = 1.50*(D1+D2)
CALL SET(.05,.95,.05,.95,-DELY,DELY,-DELY,DELY,1)
CALL CONREC (A(1,4),MA,MA,NA,0.,0.,0.,1,-1,0)
CALL FRAME
Frame 3: CONREC + EZMAP + FX + FY (Full Globe)

MA = 55
NA = 45
CALL GENARA(B,A,MA,NA)
ITRANS = 2
CALL SUPMAP (2,40.,-90.,0.,0.,0.,0.,0.,1,20,4,0,IERS)
ALNMN = -160.
ALNMX = -20.
ALTMN = 0.
ALTMX = 60.
CALL CONREC (A,MA,MA,NA,0.,0.,0.,1,0,-585)
CALL FRAME
Frame 4: CONREC + EZMAP + FX + FY (U.S.A.)

MA = 55
NA = 45
CALL GENARA(B,A,MA,NA)
ITRANS = 2
CALL SUPMAP (3,45.,-100.,45.,50.,-130.,20.,-75.,2,10,3,1,IERS)
ALNMN = -120.
ALNMX = -75.
ALTNMN = 25.
ALTMX = 50.
CALL CONREC (A,MA,MA,NA,0.,0.,0.,1,-1,1)
CALL FRAME
Frame 5: CONREC + FX + FY (Rectangular Projection)

ITRANS = 0
CALL GENARA(B,A,40,40)
CALL SET(.1,.9,.1,.9,-1.,1.,-2.,2.,1)
CALL LABMOD ('(F3.0)','(F3.0)',3,3,0,0,0,0,0)
CALL PERIML(2,10,4,5)
CALL CONREC(A(1,1),40,40,40,-1.,1.,.5,-1,-1,-82)
CALL FRAME
Frame 6: VELVCT + EZMAP + FX + FY

CALL GENARA(B,A,60,60)
ITRANS = 2
CALL SUPMAP (3,45.,-100.,45.,-130.,20.,-75.,2,15,3,0,IERS)
ALNMN = -120.
ALNMX = -75.
ALTMN = 25.
ALTMX = 50.
MA = 25
NA = 25
CALL VELVCT (A(1,10),60,A(1,25),60,MA,NA,0.,0.,1,0,0,0.)
CALL FRAME
Deactivate and close workstation, close GKS.

CALL GDAWK (1)
CALL GCLWK (1)
CALL GCLKS
STOP
END

FUNCTION FX(X,Y)

COMMON /TRANS/ MA, NA, ITRANS, A1, A2, D1, D2, ALNMN, ALNMX, ALTMN, ALTMX

This function implements three transformations on the input coordinate pair (X,Y). The coordinate (X,Y) is assumed to be contained in the rectangle 1 to MA by 1 to NA. The transformation selected depends on the value of ITRANS:

ITRANS=0 The identity transformation is used.

ITRANS=1 Polar transformation. The rectangular region bounded by the corner points (1,1), (MA,1), (MA,NA), (1,NA) is transformed onto the region in polar coordinates bounded by the polar coordinates (D1,A1), (D1,A2), (D2,A2), (D1,A2) where the angles A1 and A1 are expressed in radians.

ITRANS=2 Provides a linear mapping of the rectangular input array onto the region ALNMN to ALNMX by ALTMN to ALTMX. The four parameters ALNMN, ALNMX, ALTMN, and ALTMX are assumed to specify two longitudes (ALNMN--minimum longitude; ALNMX--maximum longitude) and two latitudes (ALTMN--minimum latitude; ALTMX--maximum latitude.) The latitudes and longitudes in the transformed space are then projected onto the plot using the current EZMAP projection (hence EZMAP must be called prior to calling FX with ITRANS=2.)

GOTO (10,20,30) ITRANS+1

Identity transformation.

10 CONTINUE
FX = X
RETURN

Polar coordinate transformation.

20 CONTINUE
FX = (D1+D2*(X-1.)/(FLOAT(MA)-1.))*COS(A1+A2*(Y-1.)/(FLOAT(NA)-1.))
RETURN
SUBROUTINE GENARA (A,B,ID,JD)
C
C This subroutine generates a smooth array in output array B.
C The array is dependent on the random number function RAND.
C RAND can be replaced by a random number generator on a
given machine, or for consistency across machines, the
supplied function RAND may be used. RAND reads its random
numbers from the file RANFDAT.
C
DIMENSION A(ID,JD), B(ID,JD)
C
DO 1 I=1,ID
DO 1 J=1,JD
1 A(I,J)=0.
August 1987 583

COMMON /TRANS/ MA, NA, ITRANS, A1, A2, D1, D2, ALNMN, ALNMX, ALTMN, ALTMX
C
The function FY behaves in an analogous manner to FX as
described above.
C
GOTO (10, 20, 30) ITRANS+1
C
The identity transformation.
C
10 CONTINUE
FY = Y
RETURN
C
Polar coordinate transformation.
C
20 CONTINUE
FY = (D1+D2*(X-1.)/(FLOAT(MA)-1.))*SIN(A1+A2*(Y-1.)/(FLOAT(NA)-1.))
RETURN
C
EZMAP overlaying.
C
30 CONTINUE
XLON = ALNMN + (X-1.)*(ALNMX-ALNMN)/(FLOAT(MA)-1.)
YLAT = ALTMN + (Y-1.)*(ALTMX-ALTMN)/(FLOAT(NA)-1.)
CALL MAPTRN (YLAT, XLON, FXLON, YDUM)
FY = FXLON
RETURN
END

FUNCTION FY(X,Y)
C
C EZMAP overlaying.
C
30 CONTINUE
XLON = ALNMN + (X-1.)*(ALNMX-ALNMN)/(FLOAT(MA)-1.)
YLAT = ALTMN + (Y-1.)*(ALTMX-ALTMN)/(FLOAT(NA)-1.)
CALL MAPTRN (YLAT, XLON, FXLON, YDUM)
FY = FXLON
RETURN
END
C

NN=(ID+JD)/10
AA=1.
DI=ID-4
DJ=JD-4
DD=MAXO(ID,JD)
C

2 DO 5 K=1,NN
   II=3.+DI*RAND()
   JJ=3.+DJ*RAND()
   DO 4 J=1,JD
      JE=IABS(J-JJ)
   DO 3 I=1,ID
      IE=IABS(I-II)
      EE=MAXO(IE,JE)
      A(I,J)=A(I,J)+AA*(.8**EE)
   3 CONTINUE
4 CONTINUE
5 CONTINUE
C

IF (AA.NE.1.) GO TO 6
AA=-1.
GO TO 2
C

6 DO 8 J=1,JD
   JM1=MAXO(1,J-1)
   JP1=MINO(JD,J+1)
   DO 7 I=1,ID
      IM1=MAXO(1,I-1)
      IP1=MINO(ID,I+1)
          +A(IM1,JM1)+A(IP1,JM1)+A(IM1,JP1)+A(IP1,JP1))/16.
    7 CONTINUE
8 CONTINUE
RETURN
END
FUNCTION RAND()
C
C This function is used to produce random numbers for the
C GENARA calls. The random numbers are read from file
C RANFDAT, and in this way consistency is maintained across
C machines (the results on one machine should be the same
C as on another.) If consistency is not required, the
C function RAND can be replaced by a local random number
C generator. RANFDAT contains 2000 random numbers in
C 250 card-image lines in format 8F10.8.
C
SAVE ICNT
DIMENSION A(2000)
DATA ICNT/O/
C
ICNT = ICNT+1
C

584 August 1987
C Read in random numbers if this is the first function call.

IF (ICNT .EQ. 1) THEN
   OPEN(8, FILE='RANFDAT')
   DO 3 I=1,250
      INDX = 8*(I-1)+1
      JNDX = INDX+7
      READ(8,501) (A(LL),LL=INDX,JNDX)
   3 CONTINUE
END IF

RAND = A(ICNT)
RETURN

C

500 FORMAT(' RAND--ERROR IN READ')
501 FORMAT(8F10.8)

C

END
Color Examples
Color Examples

AUTOGRAPH

Important Note: The AUTOGRAPH scheme for setting color requires you to supply your own versions of certain routines in the distributed package. (The default versions of these routines do nothing.) The code below includes replacement versions of three such routines: AGCHAX (to change the color of the axes), AGCHCU (to change the color of the curves), and AGCHIL (to change the color of the informational labels). If AUTOGRAPH is loaded from a binary library, compiling the replacement versions will typically be sufficient to cause them to be used in place of the default versions. If a binary library is not used, you may have to take steps to delete the default versions from the source code.

```
SUBROUTINE TAUTOC (IERR)
C
C PURPOSE
C To provide a color demonstration of the AUTOGRAPH package.
C
C USAGE
C CALL TAUTOC (IERROR)
C
C ARGUMENTS
C IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise
C
C I/O
If the test is successful, the message
COLOR AUTOGRAPH TEST EXECUTED--SEE PLOT TO CERTIFY
is printed on unit 6. In addition, 1 frame is produced on the machine graphics device. In order to determine if the test was successful, it is necessary to examine the plot.
C
C PRECISION
Single
C
C LANGUAGE
FORTRAN 77
C
C REQUIRED ROUTINES
AUTOGRAPH, DASHCHAR
C
C REQUIRED GKS LEVEL
OA
C
C ALGORITHM
TAUTOC defines a set of RGB triples for the 15 curves in the output plot.
C
C Define the data arrays.
C
REAL XDRA(201), YDRA(201,10)
```

August 1987
DIMENSION IOC(15)

C Define the RGB triples needed below.
C
DIMENSION RGB(3,15)

C DATA RGB / 0.70 , 0.70 , 0.70 ,
+ 0.75 , 0.50 , 1.00 ,
+ 0.50 , 0.00 , 1.00 ,
+ 0.00 , 0.00 , 1.00 ,
+ 0.00 , 1.00 , 1.00 ,
+ 0.00 , 1.00 , 0.60 ,
+ 0.00 , 1.00 , 0.00 ,
+ 0.70 , 1.00 , 0.00 ,
+ 1.00 , 1.00 , 0.00 ,
+ 1.00 , 0.75 , 0.00 ,
+ 1.00 , 0.38 , 0.38 ,
+ 1.00 , 0.00 , 0.38 ,
+ 1.00 , 0.00 , 0.00 ,
+ 1.00 , 1.00 , 1.00 /

C Define a set of indices determining the ordering of the colors.
C
DATA IOC / 6,2,5,12,10,11,1,3,4,8,9,7,13,14,15 /

C Define 15 different color indices. The first 14 are spaced throughout
C the color spectrum and the final one is white.
C
CALL SETUSV ('IM',15)

C DO 101 J=1,15
I=IOC(J)
IF (RGB(1,I).GE.RGB(2,I).AND.RGB(1,I).GE.RGB(3,I)) THEN
IND=1
IND=2
ELSE IF (RGB(3,I).GE.RGB(1,I).AND.RGB(3,I).GE.RGB(2,I)) THEN
IND=3
END IF
CALL SETUSV ('IR',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
CALL SETUSV ('IR',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
CALL SETUSV ('IR',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
101 CONTINUE
C
C Fill the data arrays.
C
DO 103 I=1,201
XDRA(I)=-1.+0.02*FLOAT(I-1)
IF (I.GT.101) XDRA(I)=2.-XDRA(I)
DO 102 J=1,10
YDRA(I,J)=FLOAT(J)*
+  $\sqrt{1.0000000001-XDRA(I)^2}/10.$
  IF (I.GT.101) YDRA(I,J)=-YDRA(I,J)
102  CONTINUE
103  CONTINUE
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Draw the graph, using EZMXY.
C
CALL EZMXY (XDRA,YDRA,201,10,201,'EXAMPLE 4 (EZMXY:S')
C
C Done.
C
WRITE(6,600)
600  FORMAT(' COLOR AUTOGRAPH TEST EXECUTED--SEE PLOT TO CERTIFY')
RETURN
C
END
SUBROUTINE AGCHAX (IFLG,IAXS,IPRT,VILS)
IF (IPRT.EQ.4.OR.IPR {T}.EQ.5) THEN
  IF (IFLG.EQ.0) THEN
    CALL SETUSV ('II',14)
  ELSE
    CALL SETUSV ('II',15)
  END IF
ENDIF
RETURN
END
SUBROUTINE AGCHCU (IFLG,KDSH)
IF (IFLG.EQ.0) THEN
  CALL SETUSV ('II',3+ABS(KDSH))
ELSE
  CALL SETUSV ('II',15)
END IF
RETURN
END
SUBROUTINE AGCHIL (IFLG,LBNM,LNNO)
CHARACTER*(*) LBNM
IF (LBNM.EQ.'T') THEN
  IF (IFLG.EQ.0) THEN
    CALL SETUSV ('II',14)
  ELSE
    CALL SETUSV ('II',15)
  END IF
ENDIF
RETURN
END
SUBROUTINE BNDARY
C
C Routine to draw the plotter-frame edge.
August 1987 591
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Draw the graph, using EZMXY.
C
CALL EZMXY (XDRA,YDRA,201,10,201,'EXAMPLE 4 (EZMXY:S')
C
C Done.
C
WRITE(6,600)
600  FORMAT(' COLOR AUTOGRAPH TEST EXECUTED--SEE PLOT TO CERTIFY')
RETURN
C
END
SUBROUTINE AGCHAX (IFLG,IAXS,IPRT,VILS)
IF (IPRT.EQ.4.OR.IPR {T}.EQ.5) THEN
  IF (IFLG.EQ.0) THEN
    CALL SETUSV ('II',14)
  ELSE
    CALL SETUSV ('II',15)
  END IF
ENDIF
RETURN
END
SUBROUTINE AGCHCU (IFLG,KDSH)
IF (IFLG.EQ.0) THEN
  CALL SETUSV ('II',3+ABS(KDSH))
ELSE
  CALL SETUSV ('II',15)
END IF
RETURN
END
SUBROUTINE AGCHIL (IFLG,LBNM,LNNO)
CHARACTER*(*) LBNM
IF (LBNM.EQ.'T') THEN
  IF (IFLG.EQ.0) THEN
    CALL SETUSV ('II',14)
  ELSE
    CALL SETUSV ('II',15)
  END IF
ENDIF
RETURN
END
SUBROUTINE BNDARY
C
C Routine to draw the plotter-frame edge.
August 1987 591
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Draw the graph, using EZMXY.
C
CALL EZMXY (XDRA,YDRA,201,10,201,'EXAMPLE 4 (EZMXY:S')
C
C Done.
C
WRITE(6,600)
600  FORMAT(' COLOR AUTOGRAPH TEST EXECUTED--SEE PLOT TO CERTIFY')
RETURN
C
END
SUBROUTINE AGCHAX (IFLG,IAXS,IPRT,VILS)
IF (IPRT.EQ.4.OR.IPR {T}.EQ.5) THEN
  IF (IFLG.EQ.0) THEN
    CALL SETUSV ('II',14)
  ELSE
    CALL SETUSV ('II',15)
  END IF
ENDIF
RETURN
END
SUBROUTINE AGCHCU (IFLG,KDSH)
IF (IFLG.EQ.0) THEN
  CALL SETUSV ('II',3+ABS(KDSH))
ELSE
  CALL SETUSV ('II',15)
END IF
RETURN
END
SUBROUTINE AGCHIL (IFLG,LBNM,LNNO)
CHARACTER*(*) LBNM
IF (LBNM.EQ.'T') THEN
  IF (IFLG.EQ.0) THEN
    CALL SETUSV ('II',14)
  ELSE
    CALL SETUSV ('II',15)
  END IF
ENDIF
RETURN
END
SUBROUTINE BNDARY
C
C Routine to draw the plotter-frame edge.
August 1987 591
CALL PLOTIT ( 0, 0.0)
CALL PLOTIT (32767, 0.1)
CALL PLOTIT (32767, 32767, 1)
CALL PLOTIT (0, 32767, 1)
CALL PLOTIT (0, 0.1)
RETURN
END

**Important Note:** The example plot is reproduced here in black and white. If the code above were used to drive a color output device, the output would be in color. The boundary, the axes, and the labels 'X' and 'Y' should be white, the numeric labels and the top label should be red, and the elliptical curves should be (from smallest to largest) pink, yellow, orange, gray, purple, blue, green, yellow-green, blue-green, and red.
EZMAP

Important Note: The EZMAP scheme for setting color requires you to supply your own version of a routine in the distributed package called MAPUSR. (The default version of this routine does nothing.) The code below includes a replacement version of MAPUSR. If EZMAP is loaded from a binary library, compiling the replacement version will typically be sufficient to cause it to be used in place of the default version. If a binary library is not used, you may have to take steps to delete the default version of MAPUSR from the source code.

SUBROUTINE TMAPC (IERR)

C
C PURPOSE To provide a color demonstration of
C the EZMAP package.
C
C USAGE CALL TMAPC (IERROR)
C
C ARGUMENTS
C ON OUTPUT IERROR
C An integer variable
C = 0, if the test was successful,
C = 1, otherwise
C
C I/O If the test is successful, the message
C COLOR EZMAP TEST EXECUTED--SEE PLOT TO CERTIFY
C
C is printed on unit 6. In addition, 1 output
C frame is produced on the machine graphics
C device. In order to determine if the test
C was successful, it is necessary to examine
C the plot.
C
C PRECISION Single
C
C LANGUAGE FORTRAN 77
C
C REQUIRED ROUTINES EZMAP, modified MAPUSR
C
C REQUIRED GKS LEVEL OA
C
C ALGORITHM TMAPC defines a set of RGB triples for the
C EZMAP continental outlines.
C
C DIMENSION IOC(15)
C
C Define the RGB triples needed below.
C
C DIMENSION RGB(3,15)
C
C DATA RGB / 0.70 , 0.70 , 0.70 ,
C + 0.75 , 0.50 , 1.00 ,
C + 0.50 , 0.00 , 1.00 ,
C Define a set of indices determining the ordering of the colors.
C
DATA IOC / 6,2,5,12,10,11,1,3,4,8,9,7,13,14,15 /
C
C Define 15 different color indices. The first 14 are spaced throughout
C the color spectrum and the final one is white.
C
CALL SETUSV ('IM',15)
C
DO 101 J=1,15
I=IOC(J)
IF (RGB(1,I).GE.RGB(2,I).AND.RGB(1,I).GE.RGB(3,I)) THEN
IND=1
IND=2
ELSE IF (RGB(3,I).GE.RGB(1,I).AND.RGB(3,I).GE.RGB(2,I)) THEN
IND=3
END IF
CALL SETUSV ('IR',IFIX(10000.*RGB(1,I)/RGB(IND,I)))
CALL SETUSV ('IG',IFIX(10000.*RGB(2,I)/RGB(IND,I)))
CALL SETUSV ('IB',IFIX(10000.*RGB(3,I)/RGB(IND,I)))
CALL SETUSV ('IN',IFIX(10000.*RGB(IND,I)))
101 CONTINUE
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDRY
C
C Set up EZMAP to use an elliptical perimeter, the "political" outline
C dataset, an orthographic projection, and maximal area.
C
CALL MAPSTI ('EL',1)
CALL MAPSTC ('OU','PO')
CALL MAPROJ ('OR',0.,-20.,0.)
CALL MAPSET ('MA',0.,0.,0.,0.)
C
C Draw the map.
C
CALL MAPDRW
C Done.

C

WRITE (6,600)
600 FORMAT(' COLOR EZMAP TEST EXECUTED--SEE PLOT TO CERTIFY')
RETURN

C

END
SUBROUTINE MAPUSR (IPRT)
IF (IPRT.GT.0) THEN
  IF (IPRT.EQ.2) CALL SETUSV ('II',9)
  IF (IPRT.EQ.5) CALL SETUSV ('II',5)
  IF (IPRT.EQ.7) CALL SETUSV ('II',13)
ELSE
  IF (IPRT.EQ.-2.OR.IPRTEQ.-5.OR.IPRTEQ.-7)
+ CALL SETUSV ('II',15)
END IF
RETURN
END

SUBROUTINE BNDRY
C
C Routine to draw the plotter-frame edge.
C
CALL PLOTIT ( 0, 0,0)
CALL PLOTIT (32767, 0,1)
CALL PLOTIT (32767,32767,1)
CALL PLOTIT ( 0,32767,1)
CALL PLOTIT ( 0, 0,1)
RETURN
END

Important Note: The example plot is reproduced here in black and white. If the code above were used to drive a color output device, the plot would be in color. The square boundary, the circular perimeter, and the labels should be white. The grid should be blue, the continental outlines should be yellow, and the political boundaries should be red.
Code for the Cover: AREAS + EZMAP + EZMAPA

The color globes on the cover were created by a program using the GKS-based NCAR Graphics Package on the CRAY,C1 computer at NCAR. Various graphics utilities described below were used to generate a metacode file, which was then sent to an IBM 4381 (IO) front-end computer. The metacode file was translated there by the NCAR plotting routine PLT for display on a TEKTRONIX 4115 terminal and photographed from the screen, using a large format camera (Hasselblad) and Kodak Ektachrome ASA 200 film.

The CRAY time required to produce this plot was about 74 seconds on the CRAY,C1 and the integer workspace required was a little more than 200,000 words. Translation of the metacode on the IBM 4381 required about 5 minutes per frame.

PROGRAM COVER

C COVER is a program that uses the GKS graphics utilities EZMAPA, AREAS, C and EZMAP to produce 3 color versions of the world map, as seen from 3 C different "satellite" positions. Other routines are from the NCAR System C Plot Package (SPPS) and the OA version of GKS.

C Define an array in which to construct the area map.
C
DIMENSION IAM(250000)

C Dimension the arrays needed by ARSCAM and ARDRLN for x/y coordinates.
C
DIMENSION XCS(10000),YCS(10000)

C Dimension the arrays needed by ARSCAM and ARDRLN for area and group
C identifiers.
C
DIMENSION IAI(10),IAG(10)

C Define the HLS color triples needed below. These will be converted to
C RGB values using a call to subroutine HLS2RGB.
C
REAL HLS(3,14), H, L, S, R, G, B
REAL RGB(3,14)
C
DATA HLS / 300., 50., 100., +
+ 140., 60., 100., +
+ 60., 85., 90., +
+ 240., 35., 70., +
+ 130., 90., 100., +
+ 30., 60., 90., +
+ 60., 90., 90., +
+ 300., 50., 100., +
+ 140., 60., 100., +
+ 60., 85., 90., +
+ 240., 35., 70., +
C Define a set of indices to order the colors.
C
DIMENSION IOC(14)
C
DATA IOC / 1,2,3,4,5,6,7,8,9,10,11,12,13,14 /
C
C Define an array for GKS aspect source flags.
C
DIMENSION IF(13)
C
C Declare the routine which will color the areas.
C
EXTERNAL COLRAM
C
C Declare the routine which will draw lines of latitude and longitude over water.
C
EXTERNAL COLRLN
C
C Open GKS.
C
CALL OPNGKS
C
C Set the aspect source flags for FILL AREA INTERIOR STYLE and for FILL AREA STYLE INDEX to "individual".
C
CALL GQASF (IE,IF)
IF(11)=1
IF(12)=1
CALL GSASF (IF)
C
C Convert HLS values to RGB values.
C
DO 999 I=1,14
H = HLS(1,I)
L = HLS(2,I)
S = HLS(3,I)
CALL HLS2RGB(H,L,S,R,G,B)
RGB(1,I)=R
RGB(2,I)=G
RGB(3,I)=B
999 CONTINUE
C
C Force solid fill.
C
CALL GSFAIS (1)
C
C Define 15 different color indices. The first 14 are spaced through the color spectrum and the final one is black.
CALL SETUSV ('IM', 15)

C

DO 101 J = 1, 14
I = IOC(J)
IF (RGB(1,I).GE.RGB(2,I).AND.RGB(1,I).GE.RGB(3,I)) THEN
  IND = 1
  IND = 2
ELSE IF (RGB(3,I).GE.RGB(1,I).AND.RGB(3,I).GE.RGB(2,I)) THEN
  IND = 3
END IF
CALL SETUSV ('IR', IFIX(10000.*RGB(1,I)/RGB(IND,I)))
CALL SETUSV ('IG', IFIX(10000.*RGB(2,I)/RGB(IND,I)))
CALL SETUSV ('IB', IFIX(10000.*RGB(3,I)/RGB(IND,I)))
CALL SETUSV ('IN', IFIX(10000.*RGB(IND,I)))
101 CONTINUE
C

CALL SETUSV ('IR', 1)
CALL SETUSV ('IG', 1)
CALL SETUSV ('IB', 1)
CALL SETUSV ('IN', 0)
C
C Set up EZMAP to use an elliptical perimeter, the "political" outline
dataset, an orthographic projection, and maximal area.
C
CALL MAPSTI ('EL', 1)
CALL MAPSTC ('OU', 'PO')
C
CALL MAPROJ ('OR', 18., -73., 8.)
CALL MAPSET ('MA', 0., 0., 0., 0.)
C
Make MAPBLA use 1 and 2 as the group identifiers.
C
CALL MAPSTI ('G1', 1)
CALL MAPSTI ('G2', 2)
C
Use 150 vertical strips to reduce the number of points defining the
C sub-areas.
C
CALL MAPSTI ('VS', 150)
C
Initialize EZMAP.
C
CALL MAPINT
C
Initialize the area map.
C
CALL ARINAM (IAM, 250000)
C
Add edges to the area map.
C
CALL MAPBLA (IAM)
C
Pre-process the area map.

August 1987 599
CALL ARPRAM (IAM,0.0,0)

C Compute and print the amount of space used in the area map.
C
ISU=250000-(IAM(6)-IAM(5)-1)
PRINT *, 'SPACE USED IN AREA MAP IS ',ISU

C Set the background color.
C
CALL GSCR (1,0,0.,0.,0.)

C Color the map.
C
CALL ARSCAM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRAM)

C In black, draw a perimeter and outline all the countries.
C
CALL SETUSV ('LW',4000)
CALL SETUSV ('II',15)
CALL MAPSTI ('LA',0)
CALL MAPSTI ('MV',1)
CALL MAPLBL
CALL MAPLOT

C Draw lines of latitude and longitude over water.
C
CALL MAPGRM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRLN)

C Advance frame #1.
C
CALL FRAME

C Do next projection; begin frame #2.
C
C Set up EZMAP to use an elliptical perimeter, the "political" outline
C dataset, an orthographic projection, and maximal area.
C
CALL MAPSTI ('EL',1)
CALL MAPSTC ('OU','PO')

C Make MAPBLA use 1 and 2 as the group identifiers.
C
CALL MAPSTI ('G1',1)
CALL MAPSTI ('G2',2)

C Use 150 vertical strips to reduce the number of points defining the
C sub-areas.
CALL MAPSTI ('VS',150)
C
C Initialize EZMAP.
C
CALL MAPINT
C
C Initialize the area map.
C
CALL ARINAM (IAM,250000)
C
C Add edges to the area map.
C
CALL MAPBLA (IAM)
C
C Pre-process the area map.
C
CALL ARPRAM (IAM,0,0,0)
C
C Compute and print the amount of space used in the area map.
C
ISU=250000-(IAM(6)-IAM(5)-1)
PRINT * , 'SPACE USED IN AREA MAP IS ',ISU
C
C Set the background color.
C
CALL GSCR (1,0,0.,0.,0.)
C
C Color the map.
C
CALL ARSCAM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRAM)
C
C In black, draw a perimeter and outline all the countries.
C
CALL SETUSV ('LW',4000)
CALL SETUSV ('II',15)
CALL MAPSTI ('LA',0)
CALL MAPSTI ('MV',1)
CALL MAPLBL
CALL MAPLOT
C
C Draw lines of latitude and longitude over water.
C
CALL MAPGRM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRLN)
C
C Advance frame #2.
C
CALL FRAME
C
C Do next projection; begin frame #3.
C
C
C Set up EZMAP to use an elliptical perimeter, the "political" outline
data set, an orthographic projection, and maximal area.
CALL MAPSTI ('EL',1)
CALL MAPSTC ('OU','PO')
CALL MAPROJ ('OR',20.,65.,-10.)
C flag
CALL MAPSET ('MA',0.,0.,0.,0.)
C Make MAPBLA use 1 and 2 as the group identifiers.
C
CALL MAPSTI ('G1',1)
CALL MAPSTI ('G2',2)
C
C Use 150 vertical strips to reduce the number of points defining the
C sub-areas.
C
CALL MAPSTI ('VS',150)
C
C Initialize EZMAP.
C
CALL MAPINT
C
C Initialize the area map.
C
CALL ARINAM (IAM,250000)
C
C Add edges to the area map.
C
CALL MAPBLA (IAM)
C
C Pre-process the area map.
C
CALL ARPRAM (IAM,0,0,0)
C
C Compute and print the amount of space used in the area map.
C
ISU=250000-(IAM(6)-IAM(5)-1)
PRINT *, 'SPACE USED IN AREA MAP IS ',ISU
C
C Set the background color.
C
CALL GSCR (1,0,0.,0.,0.)
C
C Color the map.
C
CALL ARSCAM (IAM,XCS,YCS,10000,IAI,IAG,10,COLRAM)
C
C In black, draw a perimeter and outline all the countries.
C
CALL SETUSV ('LW',4000)
CALL SETUSV ('II',15)
CALL MAPSTI ('LA',0)
CALL MAPSTI ('MV',1)
CALL MAPLBL
CALL MAPLOT
C
C Draw lines of latitude and longitude over water.
C
CALL MAPGRM (IAM, XCS, YCS, 10000, IAI, IAG, 10, COLRLN)
C
C Advance frame #3.
C
CALL FRAME
C
C Close GKS.
C
CALL CLSGKS
C
C Done.
C
STOP
C
END
C
C Define the routine to color the various areas of the map.
C
SUBROUTINE COLRAM (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
ITM=1
DO 101 I=1, NAI
   IF (IAI(I).LT.0) ITM=0
101 CONTINUE
IF (ITM.NE.O) THEN
   ITM=0
   DO 102 I=1, NAI
      IF (IAG(I).EQ.1) ITM=IAI(I)
102 CONTINUE
IF (ITM.GT.0) THEN
   IF (NCS.GT.150) PRINT *, 'COLRAM - NCS TOO BIG - ',NCS
   CALL SETUSV ('II', MAPACI (ITM))
   CALL GFA (NCS-1, XCS, YCS)
END IF
END IF
RETURN
END
C
C Define the routine to color the various areas of the map.
C
SUBROUTINE COLRLN (XCS, YCS, NCS, IAI, IAG, NAI)
DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
ITM=1
DO 101 I=1, NAI
   IF (IAI(I).LT.0) ITM=0
101 CONTINUE
IF (ITM.NE.O) THEN
   ITM=0
   DO 102 I=1, NAI
      IF (IAG(I).EQ.1) ITM=IAI(I)
102 CONTINUE
   END IF
END IF
RETURN
END
102 CONTINUE
IF (MAPACI(ITM).EQ.1) THEN
   IF (NCS.GT.150) PRINT *, 'COLRLN - NCS TOO BIG - ',NCS
   CALL SETUSV ('II',15)
   CALL GPL (NCS,XCS,YCS)
END IF
END IF
RETURN
END

SUBROUTINE HLS2RGB(H,L,S,R,G,B)

C
C Convert HLS to RGB color coordinate system; on entry H is in [0,360],
C L and S are in [0,100]; on exit R,G,B are in [0,1].
C
REAL L,M1,M2
L = L/100.
S = S/100.
IF (S .EQ. 0.0) THEN ! achromatic; there is no hue
   R = L
   G = L
   B = L
ELSE ! chromatic; there is a hue
   IF (L .LE. 0.5) THEN
      M2 = L * (1.0 + S)
   ELSE
      M2 = L + S - (L * S)
   ENDIF
   M1 = 2.0 * L - M2
   R = RGBVAL(M1,M2,H)
   G = RGBVAL(M1,M2,H - 120.0)
   B = RGBVAL(M1,M2,H + 120.0)
ENDIF
RETURN
END

FUNCTION RGBVAL(M1,M2,HUE)
REAL M1,M2
H = HUE
IF (H .LT. 0.) H = H + 360.
IF (H .GT. 360.) H = H - 360.
IF (H .LT. 60.0) THEN
   RGBVAL = M1 + (M2 - M1) * H / 60.0
ELSE IF (H .LT. 180.0) THEN
   RGBVAL = M2
ELSE IF (H .LT. 240.0) THEN
   RGBVAL = M1 + (M2 - M1) * (240.0 - H) / 60.0
ELSE
   RGBVAL = M1
ENDIF
IF (RGBVAL.GT.1.) RGBVAL = 1.
IF (RGBVAL.LT.0.) RGBVAL = 0.
RETURN
END
SPPS Test
The following FORTRAN code provides a sample test driver for the SPPS portion of NCAR graphics. It calls a series of non-plot entries including SET, GETSET, FL2INT, SETI, GETSI, FRSTPT, and MXMY. If the calls are successful a printout to that effect is generated. This test package then calls the following plotting entries of SPPS: PWRIT, WTSTR, PLOTIT, LINE, CURVE, VECTOR, SET, POINT, and POINTS.

```fortran
BLOCKDATA
COMMON /BLOCK1/MESG,IDUMMY(500)
C
C Set output unit for messages.
C
DATA MESG/6/
END
SUBROUTINE TSPPS (IERR)

C PURPOSE
To provide a simple demonstration of the more commonly used SPPS entries.

C USAGE
CALL TSPPS (IERROR)

C ARGUMENTS
C
C ON OUTPUT
IERROR
An integer variable
= 0, if the test was successful,
= 1, otherwise

C I/O
If the test is successful, the message
SPPS TEST EXECUTED--SEE PLOTS TO CERTIFY
is printed on unit 6. In addition, 8 frames are produced on the machine graphics device. In order to determine if the test was successful, it is necessary to examine the plots.

C PRECISION
Single

C LANGUAGE
FORTRAN 77

C REQUIRED ROUTINES
SPPS

C REQUIRED GKS LEVEL
OA

COMMON /BLOCK1/MESG,IDUMMY(500)

C IERR=0
C
C Use the SPPS entry to open GKS.
```
CALL OPNGKS

C Check the non-plotting entries.

C

C SET and GETSET (SET is also tested below.)

CALL SET(0., 1., 0., 1., 0., 1., 10., 100., 1)
CALL GETSET(PXA, PXB, PYA, PYB, XC, XD, YC, YD, LTYPE)
IF (PXA .NE. 0.0 .OR. PXB .NE. 1.0 .OR.
    PYA .NE. 0.0 .OR. PYB .NE. 1.0 .OR.
    XC .NE. 0. .OR. XD .NE. 1. .OR.
    YC .NE. 10. .OR. YD .NE. 100. .OR. LTYPE .NE. 1)
    GO TO 200
WRITE (MESG, 500)
GO TO 210

200 CONTINUE
WRITE (MESG, 510)

C IERR=1
C RETURN
210 CONTINUE

C FL2INT

CALL SET(0., 1., 0., 1., 0., 1., 1)
CALL FL2INT(O., O., MX, MY)
IF (MX .NE. 0 .OR. MY .NE. 0) GO TO 221

CALL FL2INT(1., 1., MX, MY)
IF (MX .NE. 32767 .OR. MY .NE. 32767) GO TO 221

CALL FL2INT(.5, .5, MX, MY)
IF (MX .NE. 16383 .OR. MY .NE. 16383) GO TO 221

WRITE (MESG, 520)
GO TO 220

221 CONTINUE

WRITE (MESG, 530)
IERR=1

220 CONTINUE

C SETI and GETSI

CALL SETI(13, 5)
CALL GETSI(LX, LY)

IF (LX .NE. 13 .OR. LY .NE. 5) THEN
WRITE (MESG, 550)
IERR=1
SPPS

GO TO 230
END IF
C
WRITE (MESG, 540)
CALL SETI(10,10)
230 CONTINUE
C
FRSTPT and MXMY
C
CALL FRSTPT(.5,.5)
CALL MXMY(MX,MY)
C
IF (MX.NE.512 .OR. MY.NE.512) THEN
WRITE (MESG, 570)
IERR=1
GO TO 240
END IF
C
WRITE (MESG, 560)
240 CONTINUE
C
Test plotting entries.
C
CALL TWTSTR
CALL TPTIAI
CALL TLINE
CALL TCURVE
CALL TVECTO
CALL TSET
CALL TPOINT
CALL TPNTS
C
Use the SPPS entry to close GKS.
C
CALL CLSGKS
C
WRITE(6,600)
600 FORMAT(' SPPS TEST EXECUTED--SEE PLOTS TO CERTIFY')
RETURN
C
500 FORMAT(' SET AND GETSET TEST SUCCESSFUL')
510 FORMAT(' SET AND GETSET TEST NOT SUCCESSFUL')
520 FORMAT(' FL2INT TEST SUCCESSFUL')
530 FORMAT(' FL2INT TEST NOT SUCCESSFUL')
540 FORMAT(' SETI AND GETSI TEST SUCCESSFUL')
550 FORMAT(' SETI AND GETSI TEST NOT SUCCESSFUL')
560 FORMAT(' FRSTPT AND MXMY TEST SUCCESSFUL')
570 FORMAT(' FRSTPT AND MXMY TEST NOT SUCCESSFUL')
C
END
SUBROUTINE BOX
CALL PLOTIT (0,0,0)
CALL PLOTIT (32767,0,1)

August 1987
The following code and associated plot show how a text field is generated using either the PWRIT or WTSTR entries.

```fortran
SUBROUTINE TWTSTR
CHARACTER *28 TITLE
COMMON /BLOCK1/ MESG, IDUMMY(500)
CALL ERNUM(O)
CALL SET(O.,1.,0.,1.,O.,1.,O.,1.,1)
TITLE(1:28)='DEMONSTRATION PLOT FOR WTSTR'
CALL WTSTR(0.5,0.9,TITLE(1:28),30,0,0)
TITLE (1:3)='AND'
CALL WTSTR(0.5,0.7,TITLE(1:3),30,0,0)
C
C The WTSTR call replaces the following PWRIT call.
C
CALL PWRIT(0.5,0.5,'DEMONSTRATION PLOT FOR PWRIT',28,30,0,0)
CALL FRAME
WRITE(MESG,740)
740 FORMAT(' TWTSTR EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
```

DEMONSTRATION PLOT FOR WTSTR

AND

DEMONSTRATION PLOT FOR PWRIT
PLOTIT

The following code and associated plot show how a square with diagonals is created using subroutine PLOTIT.

SUBROUTINE TPLMTI
COMMON /BLOCK1/ MESP, IDUMMY(500)
CALL FNUM(0)
CALL SET(0,1,0,1,0,1,0,1,1)
CALL WTSTR(0.5,0.91,'DEMONSTRATION PLOT FOR PLOTIT',20,0,0)
CALL PLOTIT (0,0,0)
CALL PLOTIT (32767,0,1)
CALL PLOTIT (32767,32767,1)
CALL PLOTIT (0,32767,1)
CALL PLOTIT (0,0,1)
CALL PLOTIT (32767,32767,1)
CALL PLOTIT (0,32767,0)
CALL PLOTIT (32767,0,1)
CALL FRAME
WRITE (MESG,710)
710 FORMAT(' TPLOTI EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
LINE

The following code and associated plot show several LINE calls that form a box with diagonals.

```fortran
SUBROUTINE TLINE
COMMON /BLOCK1/ MESG, IDUMMY(500)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL FRNUM(0)
CALL WTSTR(0.5,0.91,'DEMONSTRATION PLOT FOR LINE',20,0,0)
CALL LINE(0.,0.,1.,0.)
CALL LINE(1.,0.,1.,1.)
CALL LINE(1.,1.,0.,1.)
CALL LINE(0.,1.,0.,0.)
CALL LINE(0.,1.,1.,0.)
CALL LINE(0.,0.,1.,1.)
CALL FRAME
WRITE(MESG,700)
700 FORMAT(' TLINE EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
```

DEMONSTRATION PLOT FOR LINE

[Image of a box with diagonals drawn by the LINE calls described in the code.]
The following code and associated plot show a CURVE call that generates a simple curve.

```fortran
SUBROUTINE TCURVE
COMMON /BLOCK1/MESG,X(51),Y(51),IDUMMY(398)
CALL BOX
CALL FRNUM(O)
CALL SET(0.,1.,0.,1.,0.,1.,0.,3,1)
CALL WTSTR(0.5,0.91,'DEMONSTRATION PLOT FOR CURVE',20,0,0)
XINC = 1./50.
DO 60 I=1,26
   X(I) = (I-1)*XINC
   Y(I) = X(I)*X(I)
60 CONTINUE
DO 70 I=1,25
   X(52-I) = 1.-(I-1)*XINC
   Y(52-I) = Y(I)
70 CONTINUE
CALL CURVE(X,Y,51)
CALL FRAME
WRITE (MESG, 640)
640 FORMAT(' TCURVE EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
```

![Demonstration Plot for Curve](image_url)
The following code and associated plot show how repeated calls to the VECTOR subroutine can generate the same curve as a single call to the CURVE subroutine.

```fortran
SUBROUTINE TVECTO
COMMON /BLOCK1/ MESSG,X(51),Y(51),IDUMMY(398)
CALL BOX
CALL FNUM(0)
CALL SET(0.,1.,0.,1.,0.,1.,0.,3,1)
XINC = 1./50.
DO 30 I=1,26
   X(I) = (I-1)*XINC
   Y(I) = X(I)*X(I)
30 CONTINUE
DO 40 I=1,25
   X(52-I) = 1.-(I-1)*XINC
   Y(52-I) = Y(I)
40 CONTINUE
   CALL FRSTPT(X(1),Y(1))
   DO 50 I=2,51
      CALL VECTOR(X(I),Y(I))
50 CONTINUE
   CALL WTSTR(0.5,0.91,'DEMONSTRATION PLOT FOR VECTOR',20,0,0)
   CALL FRAME
   WRITE (MESSG, 630)
630 FORMAT(' TVECTO EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
```

![Demonstration plot for vector](image)
The following code and associated plot show how a SET call establishes the boundary of the plot window.

```
SUBROUTINE TSET
  COMMON /BLOCK1/ MESG, IDUMMY(500)
  CALL FRNUM(0)
  CALL SET(0., 1., 0., 1., 0., 1., 0., 1., 1)
  CALL WTSTR(0.5, 0.975, 'DEMONSTRATION PLOT FOR SET', 25, 0, 0)
  DO 140 I = 1, 10
    UMIN = .05*(I-1)
    UMAX = 1.-UMIN
    CALL SET(UMIN, UMAX, UMIN, UMAX, 0., 1., 0., 1., 1)
    CALL PERIM(1, 1, 1, 1)
  140 CONTINUE
  CALL FRAME
  WRITE (MESG, 770)
  770 FORMAT(' TSET EXITED--SEE PLOT TO VERIFY PERFORMANCE')
  RETURN
END
```

DEMONSTRATION PLOT FOR SET
POINT

The following code and associated plot demonstrate repeated calls to POINT.

```fortran
SUBROUTINE TPOINT
  COMMON /BLOCK1/ MESC,IDUMMY(500)
  CALL ERNUM(0)
  CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
  CALL WTSTR(0.5,0.91,'DEMONSTRATION PLOT FOR POINT',20,0,0)
  DO 120 I=1,512
    IX = I*2
    IY = I*2
    CALL POINT(FLOAT(IX)/1024.,FLOAT(IY)/1024.)
    IY = 1024-IY
    CALL POINT(FLOAT(IX)/1024.,FLOAT(IY)/1024.)
  120 CONTINUE
  CALL FRAME
  WRITE(MESC,720)
  720 FORMAT( ' TPOINT EXITED--SEE PLOT TO VERIFY PERFORMANCE')
  RETURN
END
```

DEMONSTRATION PLOT FOR POINT
The following code and associated plot show how a POINTS call generates a series of points with connecting lines.

SUBROUTINE TPNTS
  COMMON /BLOCK1/ MESSG, X(11), Y(11), IDUMMY(478)
  CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
  CALL FRNUM(0)
  CALL WTSTR(0.5,0.91,'DEMONSTRATION PLOT FOR POINTS',20,0,0)
  CALL BOX
  DO 130 I=1,9
    X(I) = .1*I
    Y(I) = .50
  CONTINUE
  ICH = ICHAR('X')
  CALL POINTS(X,Y,9,ICH,1)
  CALL FRAME
  WRITE (MESSG, 730)
  730 FORMAT(' TPNTS EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
GKS Level 0A Output Primitives
The following FORTRAN code and plot demonstrate the five GKS output primitives: POLYLINE, POLYMARKER, TEXT, FILL AREA, and CELL ARRAY.

```
PROGRAM GKSOA
GKS sample program, level 0A.
Puts out sample POLYLINE, POLYMARKER, TEXT, FILL AREA and CELL ARRAY.
(Knowledge of device and environment assumed -- normal inquiry and
error checking are omitted from the sample program).

REAL ZZX(9), ZZYL(9), ZZYM(9), CIRX(9), CIRY(9)
INTEGER ICELLS(24,12)

DATA ZZX/ -9.0, -8.0, -7.0, -6.0, -5.0, -4.0, -3.0, -2.0, -1.0/ 
DATA ZZYL/ 6.5, 8.5, 6.5, 8.5, 6.5, 8.5, 6.5, 8.5, 6.5/ 
DATA ZZYM/ 1.5, 3.5, 1.5, 3.5, 1.5, 3.5, 1.5, 3.5, 1.5/ 
DATA CIRX/ 6.15, 5.26, 4.25, 3.59, 6.99, 6.20, 6.02, 6.54, 7.50/ 
DATA CIRY/ 8.46, 8.98, 8.80, 8.01, 6.99, 6.20, 6.02, 6.54, 7.50/
DATA ISZ/ 0/

CALL GOPKS (6,ISZ)  
Open GKS w/ logical unit 6 for errors.

CALL GOPWK (1, 2, 1)  
Open a wksnt of type 1, assigning 1 for its id, 2 for connection id.

CALL GACWK (1)  
Activate the workstation.

CALL GSWN (1, -10.0, 10.0, -10.0, 10.0)  
Define normalization transformation 1. Window in World Coordinates.

CALL GSVP (1, 0.1, 0.9, 0.1, 0.9)  
Viewport in Normalized Device Coords.

CALL GSELNT (1)  
Select just-defined transformation.

CALL GPL (9, ZZX, ZZYL)  
Draw a zig-zag POLYLINE.

"+"  
Set marker type.

CALL GSMK (2)  
Draw a POLYMARKER.

CALL GPM (9, ZZX, ZZYM)
```

August 1987
C C C solid Set fill area interior style.
CALL GSFAIS (1)
C C C C Draw a solid circle w/FILL AREA.
CALL GFA (9, CIRX, CIRY)
C C C Define 24x12 foreground/background checkerboard pattern.
DO 25 IX=1,24
  JX = MOD(IX,2)
  DO 20 IY=1,12
    JY = MOD(IY,2)
    IF ((JX.EQ.1 .AND. JY.EQ.1) .OR. (JX.EQ.0 .AND. JY.EQ.0)) THEN
      ICELLS (IX,IY) = 1
    ELSE
      ICELLS (IX,IY) = 0
    ENDIF
  CONTINUE
CONTINUE
C C Draw the checkerboard w/ CELL ARRAY.
CALL GCA (2.,1.5,8.,4.0, 24, 12, 1, 1, 24, 12, ICELLS)
C C C Set chr hgt to 2% of screen (.02*20).
CALL GSCHH (0.4) cen,hlf
C C C Set alignment to center string on posn.
CALL GSTXAL (2, 3)
XPOS = 0.0
YPOS = -5.0
C C Draw the text string.
CALL GTX (XPOS, YPOS, 'Example string, centered in the center')
C C C Flush metacode buffer.
CALL GCLRWK(1,1)
C C C Deactivate the workstation.
CALL GDAWK (1)
C C C Close the workstation.
CALL GCLWK (1)
C C C Close GKS
C C WRITE (6,600)
600 FORMAT(' GKSOA TEST EXECUTED--SEE PLOT TO CERTIFY')
STOP
END

August 1987
Example string, centered in the center
Computer Graphics Metafile (CGM)
Plot with Hex and Octal Dumps
Computer Graphics Metafile (CGM) Plot with Hex and Octal Dumps

The following FORTRAN code and subsequent plot show a test program that generates a simple metafile. The contents of the metafile should agree with the contents shown in the code listing.

SUBROUTINE TCGM (IERR)
C
C This program produces a small plotting instruction stream.
C The primary purpose of this program is to test a local
C implementation of the NCAR GKS OA package, including PLOTIT
C of SPPS. Only a trivial subset of the package is used in
C this minimal test. Execution of this program using NCAR's
C GKS package will produce a CGM. Hex and octal dumps of this
C CGM follow:
C
C In HEX:
C
C Record 1:
C 001E 3400 0021 0000 1022 8001 104B 0A4E
C 4341 525F 474B 5330 4100 1166 0001 FFFF
C 0000 0000 0000 0000 0000 0000 0000 0000
C .
C .
C .
C (86 rows of zeros as in the above line.)
C .
C .
C 0000 0000 0000 0000 0000 0000 0000 0000
C

Record 2:
C 005C 3800 0061 0000 0080 30C2 0001 4034
C 0000 0000 7FFF 0000 7FFF 7FFF 0000 7FFF
C 0000 0000 524C 0002 0003 0000 0000 0000
C 0000 51E2 02B5 5208 0000 02D5 02D5 0000
C 409C 3FFF 3FFF 0000 154E 4341 5220 474B
C 5320 5061 636B 6167 6520 5465 7374 00A0
C 0000 0000 0000 0000 0000 0000 0000 0000
C .
C .
C .
C (82 rows of zeros as in the above line.)

August 1987 627
only the first 6 bytes of record 3 are significant, and there
may be non-zero values in the remaining 1434 bytes of that
record due to the fact that the metafile output buffer has
not been cleared from previous writes (clearing is not
necessary since the record contains a count of how many
bytes are significant in the record.)

In OCTAL bytes:

Record 1:
000 036 064 000 000 041 000 000 020 042 200 001 020 113 012 116
103 101 122 137 107 113 123 060 101 000 021 146 000 001 377 377
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
.
.
.
(86 rows of zeros as in the above line.)
.
.
.

000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
.
.
.

Record 2:
000 134 070 000 000 141 000 000 000 200 060 302 000 001 100 064
000 000 000 000 177 377 000 000 177 377 177 377 000 000 177 377
000 000 000 000 122 114 000 002 000 003 000 000 000 000 000 000
(82 rows of zeros as in the above line.)

(87 rows of zeros as in the previous line.)

Only the first 6 bytes of record 3 are significant, and there
may be non-zero values in the remaining 1434 bytes of that
record due to the fact that the metafile output buffer has
not been cleared from previous writes (clearing is not
necessary since the record contains a count of how many
bytes are significant in the record.)

DATA N/32767/

Initialize the error condition.

IERR=0

TRIVIAL FRAME

CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PLOTIT (0.,0.,0)
CALL PLOTIT (N.,0.,1)
CALL PLOTIT (N.,N.,1)
CALL PLOTIT (0.,N.,1)

August 1987
CALL PLOTIT (0,0,1)
CALL WTSTR(.5,.5,'NCAR GKS Package Test',3,0,0)
CALL FRAME
C
WRITE (6,600)
600 FORMAT(' CGM TEST EXECUTED--SEE PLOT TO CERTIFY')
RETURN
END

NCAR GKS Package Test
Index

A
Absolute integer addressing. See Coordinate addressing
Application programs, 9-10, 11
Area. See AREAS
Area identifiers, 231, 271-286, 405-411, 479-484, 597-604. See also AREAS
Area maps, 7, 8, 65, 71-87
   examples, 405-411, 479-484, 597-604
   steps to prepare, 76-79
   See also AREAS; Maps
AREAS, 7, 8, 63, 64, 65, 71-87
   error messages, 86-87
   examples, 405-411, 597-604
Argument lists. See individual utilities
Arrows, 8, 9, 66, 67. See also STRMLN; VELVCT
AUTOGRAPH, 5, 7, 14, 63, 64, 65, 91-103
   examples, 413-419, 573, 575-576, 589-592
   See also DASHCHAR; DASHSMTH
AUTOGPH. See AUTOGRAPH
Automatic graphing. See AUTOGRAPH; Graphs
Axes. See AUTOGRAPH; GRIDAL
Azimuthal projection, 208, 242, 471-478

B
Backgrounds. See AUTOGRAPH; GRIDAL
Bar charts. See HISTGR
Buffer, 12, 29, 51, 54
   flush, 48, 51, 52, 56, 206. See also FLUSH

C
Centering, 52, 56
CGM. See Computer Graphics Metafile (CGM)
Characters, 28, 60, 64, 455.
   alignment, 53, 56
   as arguments to subroutines, 40
   color, 52
   concatenation operator in FORTRAN, 41
   drawing in three space, 8, 9, 66. See also PWRZI, PWRZS, PWRZT
   fancy. See Fancy characters
   manipulation, 27, 40
   orientation, 52, 53, 56
   sizes, 23, 24, 46, 53, 56, 456, 460, 464, 552
   software, 8, 66, 527-533, 535-537
   strings, 32, 45, 52, 53, 56
variables, 21, 37-41
See also PWRIT; PWRTM; PWRITX; PWRITY; PWRZI; PWRZS; PWRZT; SRFACE;
    THREED; WTSTR
    TYPE CHARACTER variable, PWRIT, WTSTR, PWRTM
Clipping, 21, 34
   CLSGKS, 13, 22, 29, 30, 33, 45, 48, 603, 609, 622
    examples, 405-411, 479-484, 505-515, 589-604
    fill, 71, 76, 85, 87
    indices, 5, 14-16, 54, 55, 59, 63, 107, 125, 139, 205, 590, 594, 597-604
    maps, 7, 8
    RGB ratio, 58
    See also AUTOGRAPH; EZMAP; EZMAPA; HISTGR
Color interface, 14-16
Computer Graphics Metafile (CGM), 6, 7, 9, 10, 12, 16, 17, 401, 627-630
Condition-access routines, 60
CONOP, 26, 27, 31, 32, 34. See also CONRAN; CONRAQ; CONRAS
CONRAN, 7, 11, 15, 26, 27, 30-31, 32, 33-34, 64, 65, 71, 107-122
    examples, 426-429, 443-446, 573, 574
    See also DASHCHAR; DASHSMTH; GRIDAL
CONRAQ, 7, 15, 27, 64, 65, 125-136.
    examples, 435-438, 574
    See also CONRAN; GRIDAL
CONRAS, 7, 15, 27, 64, 65, 139-154
    examples, 451-454, 574
    See also CONRAN; DASHSUPR; GRIDAL
CONREC, 7, 15, 35, 64, 65, 71, 157-161, 439
    examples, 421-425, 429, 438, 439-442, 446, 454, 573, 574, 575, 577-580
    See also DASHCHAR; DASHSMTH
CONRECQCK, 7, 15, 64, 65, 165-168
    examples, 431-434, 574
    See also CONREC; DASHCHAR; GRIDAL
CONRECSMTH, 21, 35
    example, 439-442
CONRECSPR, 7, 15, 64, 65, 171-175
    examples, 447-450, 574
    See also CONREC; DASHSUPR; GRIDAL
Continental maps. See AREAS; EZMAP; EZMAPA; Maps
Contour data fields, 7
Contouring, 6, 7, 65, 71, 401
    examples, 421-454, 426-429, 431-454
    See also CONRAN; CONRAQ; CONRAS; CONREC; CONRECQCK; CONRECSPR
Contour lines, 7, 31, 34, 64, 65, 71, 84
    See also CONRAN; CONRAQ; CONRAS; CONREC; CONRECQCK; CONRECSPR; Lines
Contour plots, 3, 7, 443-446
    See also CONRAN; CONRAQ; CONRAS; CONREC; CONRECQCK; CONRECSPR
Conversion function. See SPPS

August 1987 632
Index

Converting from FORTRAN 66 to FORTRAN 77. See FORTRAN
Converting to NCAR Graphics, 6, 7, 21-35, 45-60
  example, 30-34
Coordinate addressing, 22-25, 27, 32
  absolute integer, 22-23, 24, 27, 32
  for characters, 23
  integer coordinates, 46, 48, 49, 53, 56-57, 60
  real coordinates, 48, 49, 53, 56-57, 60
  user coordinates. See Coordinate addressing, world coordinates
  user space. See Coordinate addressing, world coordinates
  on a virtual screen, 23
  world coordinates, 22, 23, 24, 27, 32, 45, 46-47, 48
Coordinate systems, 46-48, 548
Countries, maps of. See EZMAP
Cross-hatch fill, 85
CURVE, 48
Curves, 7, 48, 65, 84,
  examples, 455-458, 459-462, 463-466, 467-470, 573, 589-592, 613, 614
  smoothed, 65, 439-442
  See AUTOGRAF; DASHCHAR; DASHLINE; DASHSMTH; DASHSUPR; SPPS; THREED
Cylindrical projection, 209-210, 242, 471-478

D
DASHCHAR, 7, 8, 15, 28, 64, 65, 179-181
  examples, 421-425, 455-458, 575, 589
DASHD, 23, 26, 27, 28, 31, 32, 33-34
Dashed lines, 7, 31, 64, 78
  examples, 416, 418, 455-458, 463-466, 467-470
  See also AREAS; DASHCHAR; DASHLINE; DASHSMTH; DASHSUPR; EZMAP; Lines, dashed
DASHLINE, 8, 15, 28, 64, 65, 185-186
  examples, 431-434, 435-438, 459-462
  See also DASHCHAR
DASHSMTH, 8, 15, 28, 35, 64, 65, 189-191
  examples, 439-442, 443-446, 463-466
  See also DASHCHAR
DASHSUPR, 8, 15, 28, 64, 65, 195-199
  examples, 447-450, 451-454, 467-470
  See also DASHCHAR
Default perimeter, 74
Defining areas, 64. See also AREAS
Demonstration drivers, 7
Device independent graphics, 10, 16
DICOMED, 55, 59, 60

E
Edges, 71-87, 405-411
  of a frame, 46, 47, 53
groups of, 72-73, 81, 82, 83, 86, 87
See also AREAS
Entry points, 6, 8-9, 23, 24-25, 28. See also individual utilities
Error conditions. See Error messages
Error messages, 12, 13-14, 17, 29
See also Graphical Kernel System (GKS), individual utilities
EZMAP, 4, 5, 8, 14, 35, 63, 64, 65, 66, 205-286
area identifiers, 271-286
and AREAS, 71, 76, 84, 87
text messages, 211
texts, 231-270, 471-478, 573, 575, 578-579, 593-596, 597-604
See also AREAS; EZMAPA
EZMAPA, 5, 7, 8, 14, 63, 64, 65, 66, 87, 289-301
texts, 479-484, 597-604
See also AREAS; EZMAP
F
Fancy characters, 6-7, 8, 66. See also PWRITX; PWRITY
Field direction. See STRMLN
Field magnitude. See STRMLN
FINDG, 7, 26, 27, 32
FLASH, 21, 25, 26, 35, 46
FLINT, 49
Floating parameters, 23
Floating-point values, 23
Floating-point variables, 27
Flow-field display, 7. See also STRMLN
FLUSH, 25, 26, 48. See also PLOTIT
FORTRAN, 3, 6, 10, 12, 26, 27, 29, 45, 51, 52. See also individual utilities
and AREAS, 76, 77, 78, 79
converting from FORTRAN 66 to FORTRAN 77, 4, 21, 22, 27, 28, 32, 37-41
information sources, 17, 22, 37
as a standard, 12, 16
Fractional coordinate system, 46, 47, 49, 51, 53, 57
and AREAS, 71, 80, 86
FRAME, 25, 26, 31, 32, 34, 49
FRSTPT, 49
Functionality, 10
of GKS, 10, 21, 34
of NCAR Graphics, 6, 21
of NSPP, 10, 21, 28
of the utilities, 3, 7-9, 63-67
G
GACOLR, 306
GETSET, 49-50
GETSI, 50
GETUSV, 50
Gnomonic projection, 208, 242, 471-478
Graph backgrounds. See GRIDAL
Graphical Kernel System (GKS), 3, 6, 12, 21, 22, 45, 63, 71, 401
  as an ANSI standard, 3, 6, 10, 12, 17
  calls to, 22, 55
  closing and deactivating, 13, 22, 29, 30, 33, 45, 48, 603, 609, 622
  as a component of NCAR Graphics, 9-10
  color interface with, 14
  and coordinate addressing. See Coordinate addressing
  entry points, 25
  error messages, 13-14, 29
  and FORTRAN, 3, 10, 12, 21
  higher-level package, 12, 29, 46
  information sources, 13, 17
  as an ISO standard, 10
  level 0A, 3, 6, 10, 11, 12, 13, 21, 29, 35, 401, 597-604
  opening and activating, 12-13, 22, 29-30, 33, 45, 49, 50, 55, 598, 607-608, 621
  output primitives examples, 5, 401, 621-623
Graph paper. See GRIDAL
Graphs, 6, 8, 64, 416, 591. See also AUTOGRAPH; GRIDAL; HISTGR
Gray-scale. See HAFTON; Halftones
GRIDAL, 8, 15, 23, 25, 46, 57, 64, 66, 305-311
  examples, 485-495, 573
GRID, 307
  example, 486
GRIDL, 308
  example, 487
Grids, 485-495. See also GRIDAL
Group identifier, 73, 76, 81, 82, 86, 479-484, 597-604

H
HAFTON, 8, 15, 64, 66, 315-319
  example, 497-503
HALFAX, 308
  example, 490
Halftones, 7, 8, 64, 66, 497-503. See also HAFTON
Hershey database, 8, 66. See also PWRITX
HISTGR, 5, 8, 14, 63, 64, 66, 323-332
  example, 505-515
Histograms, 8, 66, 505-515. See also HISTGR
Holes in an area, 86
Hollerith "nH," 22, 32, 37, 45
HSTGRM. See HISTGR

I
Integer variables. See TYPE CHARACTER variable
Intensity, 5, 54-55, 58-59, 63, 78, 80, 107, 125, 139, 205,
    See also Color; CONRAN; CONRAQ; CONRAS; HAFTON; GRIDAL; SPPS
International Standards Organization (ISO), 3, 10
INTT function, 23, 26, 28
Islands. See EZMAP; Maps
ISO. See International Standards Organization (ISO)
ISOSRF, 8, 15, 64, 66, 335-338
    example, 517-521, 539-541, 573, 574
    See also PWRZI
ISOSRFHR, 8, 15, 64, 66, 341-343
    example 523-526
Iso-surfaces. See ISOSRF; ISOSRFHR

L
Labeling, 7, 8, 65, 78, 455, 487-495, 589-592. See also CONRAN; CONRAQ; CONRAS;
    CONREC; CONRECSUPR; DASHCHAR; PWRIT; PWRITX; PWRITY
LABMOD, 308-310
    example, 492
Lakes. See EZMAP
Lambert projection, 232, 208, 242, 471-478
Left area identifier, 74, 75, 76, 77, 82, 84, 231. See also Area identifiers
LINE, 11, 25, 26, 31, 32, 33, 46, 47, 50, 57, 60
Lines, 8, 49, 50, 56, 64, 65, 71, 86
    absolute lengths, 23-24
    colored, 14, 15-16, 58, 78, 80
    contoured, 7, 78, 84
    crowded, 8, 7, 65. See also CONRAS; CONRECSUPR; DASHSUPR
dashed, 8, 65, 78, 80, 416, 455-458, 463-466, 467-470. See also CONRECSUPR;
    DASHCHAR; DASHLINE; DASHSMTH; DASHSUPR; EZMAP
of flow, 8, 66
of latitude and longitude, 8, 66, 231, 479-484, 561-563, 582, 598. See also AREAS;
    EZMAP; EZMAPA
on maps, 8, 66, 71, 72, 73, 84. See also EZMAP; EZMAPA
masked, 71, 75. See also AREAS; EZMAPA
processing, 79, 80
removed, 8, 66. See also ISOSRF; ISOSRFHR; SRFACE
smoothed, 7, 8, 65. See also CONRECSUPR; DASHSUPR; DASHSMTH
in three space, 6, 8-9, 66. See also SPPS; THREED
width of, 23-24, 54, 55, 60
    See also CONRAN; CONRAQ; CONRAS; CONREC; CONRECSUPR; CONRECQCK;
    DASHSMTH; DASHSUPR; EZMAPA; THREED
Log scaling, 21-22, 31, 32, 33, 54
    automatic, 21-22
    examples, 413, 535-537
    and SET. See SET
Lower level subroutines. See Utilities, lower level
Maps, 64, 65.
  area, 7, 8, 65
colored, 7, 64, 65, 66, 67, 87, 271, 479-484. See also EZMAPA
continental, 8, 66, 483
coordinates on, 47-48, 49-50
linear, 52, 53, 305-306
logarithmic, 53, 305-306
parameters of, 86
U.S., 8, 66, 72, 73, 483, 597-604
utilities for. See also AREAS; EZMAP; EZMAPA
values on, 56-57
world political, 8, 6, 66, 483, 593-596, 597-604
See also AREAS; EZMAP; EZMAPA
Marker size, 54, 55, 60, 621
Masking, 75, 77, 78-79. See also Lines, masked
Mercator projection, 236, 239, 242, 471-478
Metacode. See Metafile
Metacode coordinate system, 46, 48, 49, 51, 60
Metacode space. See Metacode coordinate system
Metafile, 6, 10, 11, 12, 16, 29, 48, 54, 57, 60. See also NCAR CGM Translator
  example plot with hex and octal dumps, 401, 627-630
  output unit, 29, 54, 60
  unit number, 45, 60
Metafile Output (MO) workstation, 6, 12, 29, 54, 55
Metafile translator. See NCAR CGM Translator
Mollweide projection, 242, 471-478
MXMY, 50

NCAR CGM Translator, 7, 11, 12, 17, 55
NCARGRAPHICS, 11
NCAR Graphics, 3, 4, 9-10, 11, 12, 21, 22, 23, 27, 28, 29
  code for, 6-7, 30-31, 32-34
  color interface for, 14-16
  components, 9-10. See also Package components
  converting to, 4, 21-35
  description of, 3,
  error messages, 13-14
  functionality, 21, 34
  history of, 16
  how to begin using, 11-13, 29-30
  information sources for, 17, 18
  installation of, 5, 17.
  NCAR System Plot Package Simulator and, 11, 21, 22, 26, 30-31
  overview of, 6-10
  standards of, 3, 10, 12, 16
NCAR Graphics User's Guide

strengths of, 6
System Plot Package Simulator and, 6, 4
utilities. See Utilities and individual utilities
NCAR System Plot Package (NSPP), 4, 10, 11, 21
calls, 21, 22, 23, 26, 32
changes to, 21, 22, 24-26, 28, 32, 34
code, 21, 30-31
converting to GKS, 21-41
entry points, 25
major differences from SPPS, 21, 34, 45-46,
similarities to SPPS, 46-47
NSPP. See NCAR System Plot Package (NSPP)
Normalized device coordinates, 71, 81, 86
See also Fractional coordinate system

O
Obsolete routines, 22, 28, 34-35, 45
Oceans. See EZMAP
OPNGKS 12-13, 22, 29-30, 33, 45, 49, 50, 55, 598, 607-608, 621
Orthographic projection, 74, 208, 242, 471-478, 599, 600, 601
Output device, 7, 9, 10, 11, 12, 15, 37

P
Package components, 9-10
PAUs. See Plotter Address Units (PAUs)
Pen-oriented devices, 51
Pen positions, 49, 50, 51, 52, 56, 206, 387
Pen widths, 60
PERIM, 310
example, 488
Perimeters, 8, 66, 74, 75, 84, 488-495
of maps, 479-484, 594, 597-604
See also AREAS; EZMAP; GRIDAL
PERIML, 310
example, 489
Planar vector components. See STRMLN
PLOTIF, 51
PLOTIT, 25, 26, 46, 48, 49, 50, 51-52, 54, 56, 57, 607, 610-611
Plotter Address Units (PAUs), 23
Plotter coordinate system, 46, 47, 49, 50, 53-54, 56, 57
Plotter frame, 46, 47, 56
POINT, 25, 26, 46, 47, 50, 51-52, 57, 616
POINTS, 25, 26, 31, 32, 33, 45, 50, 52, 57-58, 60, 617
Polar coordinate transformation, 573, 582, 583
Political maps, 8, 594, 597-604. See also EZMAP
Polyline, 48, 51, 52, 55, 78, 80, 81
Portability of NCAR Graphics, 6, 16, 28
Potential growth of NCAR Graphics, 6. See also individual utilities
PWRIT, 23, 25, 26, 46, 50, 57, 60, 607-610
PWRITX, 8, 15, 23, 26, 27, 64, 66, 347-352
   examples, 27, 527-533
PWRITY, 8, 15, 26, 64, 66, 355-356
   example, 535-537
PWRM, 26, 28
PWRTM. See PWRM
PWRZI, 8, 15, 64, 66, 359-360. See also ISOSRF
   example, 539-541
PWRZS, 8, 15, 64, 66, 363-364. See also SRFACE
   example, 543-545
PWRZT, 8, 9, 15, 64, 66, 367-368. See also THREED
   example, 547-549

R
Regular rectangular lattice, 66. See also STRMLN
Resolution, 23-24, 341
Right area identifier, 74, 75, 76, 77, 82, 84, 231. See also Area identifiers
Routines. See also individual utilities
   obsolete, 34-35

S
SCROLL, 21, 35
SET, 21-22, 23, 25, 26, 31, 32, 33, 34, 46, 47, 48, 49-50, 52, 53, 54, 57, 71
   and AREAS, 81, 86
   description of, 53
   examples, 607, 608, 615
SETI, 23, 25, 26, 46, 53-54, 608
SETUSV, 54
Shading, 71. See also HAFTON; Halftones
Spot size, 60
SPSS. See System Plot Package Simulator (SPSS)
SRFACE, 8, 15, 16, 64, 66, 371-375. See also PWRZS
   examples, 543-545, 551-555
Smoothing, 7, 8. See also CONRAS; CONRECSPR; DASHSMTH; DASHSUPR
Solid-colored maps, 7, 8. See also AREAS; EZMAPA
Source code, 4, 6
States, U.S., See AREAS; EZMAP; Maps
Stereographic projection, 74, 208, 234, 239, 242, 471-478
Streamlines. See STRMLN
STRMLN, 8, 15, 64, 66, 379-383
   examples, 557-559, 573, 574
SUPMAP, 21, 35, 395-397
Support routines, 11, 28, 29,
   obsolete, 22, 28, 33
System Plot Package Simulator (SPPS), 4, 6, 9-10, 11, 13, 21-33, 43-60, 66, 71, 401
conversion function, 24-25, 27, 32
description of, 45-60
major differences between NSPP and, 21, 24-25, 45-46
similarities to NSPP, 46-47
subroutines of, 48-56
test plot examples, 401, 597-604, 607-617
See also individual utilities

T
Test drivers. See Demonstration drivers
Text, 52, 53, 55, 56, 58, 59
colored, 14, 15, 58, 59
Tick marks, 491
TICKS, 310
TICK4, 311
example, 491
THREED, 8-9, 15, 64, 66, 387-389
examples, 547-549, 561-563
See also PWRZT
Three-dimensional,
arrays, 8, 66. See also ISOSRF; ISOSRFHR
displays, 8, 64, 66. See also ISOSRF; ISOSRFHR; SRFACE; THREED
examples, 539-541, 547-549
line drawing, 6
Three space, 8, 9
drawing characters in. See ISOSRF; ISOSRFHR; PWRZI; PWRZS; PWRZT;
SRFACE; THREED
Transforming coordinate output, 573-585
Translator. See NCAR CGM Translator
Two-dimensional,
arrays, 7, 8, 65, 66. See also CONREC; HAFTON; SRFACE
examples, 543-545, 547-549
planes, 7, 65, 71. See also AREAS
vector field, 8, 66. See also STRMLN
velocity field, 9, 67. See also STRMLN; VELVCT
See also VELVCT
TYPE CHARACTER variable, 22, 27, 28, 37-41, 348
example, 38-41

U
User aids, 7. See also FINDG
User coordinate system, 23, 45, 46-47, 48, 49, 50, 51, 53, 56, 561-563
and AREAS, 71, 72, 76, 78, 79, 80, 82, 83
User plane, 76
User state variables, 45, 50, 53, 54
U.S. maps. See also AREAS; EZMAP; EZMAPA; Maps
Utilities, 3, 4, 5, 6-9, 10, 25, 27
  brief descriptions of, 3, 7-9, 64-67
  capabilities of, 7-9, 63-67
  color examples, 405-411, 479-484, 509-515, 589-604
  color interface with, 14-16
  as components of the package, 8, 10
  error messages, 13-14
  higher level, 6-9, 11, 12, 21, 22, 26-27, 32, 71, 76, 401
  and log scaling. See Log scaling
  lower level, 11, 21, 22
  obsolete, 21, 34-35
  See individual utilities

V
VECTOR, 25, 26, 49, 50, 56, 57, 607, 614
Vectors. See VELVCT
VELVCT, 9, 15, 64, 67, 393-397
  examples, 565-569, 573, 574, 575, 581-585
  See also GRIDAL
Viewport, 31, 33, 49, 53
Virtual screen, 23-24

W
WINDOW, 21, 23, 34
Windows, 49-50, 53
Workstation, 6, 22, 29, 45, 55
  closing and deactivating, 13, 29, 30, 33, 34, 45, 48, 582, 622
  opening and activating, 12-13, 29-30, 33, 45, 49, 50-51, 55, 575, 621
World map projections. See AREAS; EZMAP; EZMAPA
WTSTR, 52, 56