The NCAR GKS-Compatible Graphics System

Fred Clare
Lofton Henderson
Stu Henderson
Barbara Horner-Miller
John Humbrecht
Dave Kennison
Preface

Over the last 18 months, the staff of the Graphics Project has been working to convert the NCAR Graphics Package to conform to the international Graphical Kernel System (GKS) standard, as adopted by the International Standards Organization (ISO) and the American National Standards Institute (ANSI). GKS includes a FORTRAN binding that specifies the names and arguments of the library of routines that comprise a standard system. The work done by Graphics Project staff includes: converting all graphics code to FORTRAN 77; converting utilities to call GKS-based routines; creating the System Plot Package Simulator (SPPS) to replace the old NCAR System Plot Package (NSPP); and implementing GKS library routines to allow software design and testing.

This is the first release of documentation for NCAR’s GKS-based Graphics Package. Both the software and the documentation included in this package have been completely revised. If you are new to the NCAR Graphics Package, you may wish to skip over the sections dealing with differences in the new software versus the old software, conversion instructions, and the like. These sections have been provided to aid users of the old NCAR Graphics Package in their conversions to the new software.

Cover Diagram

The diagram shown on the cover of this text displays the components of the old and new packages side-by-side. The applications layer, shown at the upper level of the top boxes, is the software supplied by a scientist or programmer. Applications programs at NCAR typically display experimental data or the output of model runs. Applications software is not part of the NCAR Graphics Package.

Below the applications layer, the old and new systems begin to diverge. The new package includes an additional bottom layer, labelled GKS, in addition to the utility layer and the SPPS layer. GKS routines can be called by application, utility or SPPS software.

The old package produced NCAR metacode only; the new one offers a choice to the user. In addition to standard Computer Graphics Metafile (CGM) instructions, the user can produce instructions for a specific graphics device; GKS also provides for input from such a device, for interactive use. GKS instructions can thus be sent directly to a graphics device, or can be stored in the form of CGM instructions for later use.
Acknowledgments

Fred Clare performed most of the editorial tasks for this manual, assisted by Bob Nicol. Many thanks are due to Alan McClellan, SCD Documentation Project Intern, who provided invaluable aid in editing, formatting, and producing this document. We also wish to gratefully acknowledge the help of John Szajgin, SCD Documentation Project Leader, in handling numerous details and problems during production.
Overview of NCAR GKS-compatible Graphics Package
Overview

Table of Contents

Introduction ............................................................... 1
Package Contents .......................................................... 2
Implementation Guide ....................................................... 3
GKS 0A ............................................................................... 3
SPPS .................................................................................... 4
Metafile Translator ............................................................ 4
Higher-level Utilities ........................................................... 5
An Overview of the NCAR GKS-Compatible Graphics Package

Introduction

This document offers a brief overview of the NCAR GKS-compatible Graphics Package. It describes the package contents (code and documentation) and how to use it.

The Graphical Kernel System (GKS) will soon be declared a standard by the International Standards Organization (ISO) and the American National Standards Institute (ANSI). GKS formalizes certain graphics functionality and specifies how this functionality is to be implemented in languages such as FORTRAN 77.

GKS is a set of basic functions for computer graphics programming that can be used by many graphics-producing applications. Use of the GKS standard has the following benefits: it allows graphics application programs to be easily transported between installations; it aids graphics applications programmers in understanding and using graphics methods; and it also provides information to device manufacturers on useful graphics capabilities. The standard defines an application-level programming interface to a graphics system.

The ultimate source for learning about GKS is, of course, the description of the proposed standard itself: Information Processing Systems Computer Graphics Graphical Kernel System, X3H3/83-25r3. You can purchase a copy of this document for $35.00 from

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

This description document is a little dense (as are most standard descriptions). Fortunately, there are several other reference documents on GKS that make it easier to understand.

The article “Realizing Graphics Standards for Microcomputers” in the February, 1983 issue of BYTE Magazine is a good introduction to Graphics Standards issues and GKS. However, this article is not a complete introduction to GKS.


Computer Graphics Programming GKS — The Graphics Standard, by G. Enderle et. al. (Springer-Verlag) is excellent and readable. This book provides a complete description of the standard, as well as many examples.

The ISO GKS can be implemented in one of nine hierarchical levels, depending on the graphical input and output capabilities. The NCAR Graphics Project has written a full FORTRAN 77 implementation of GKS at Level 0A (zero A), the lowest level of GKS functionality in the ISO standard. Level 0A does not provide graphical input functions. The proposed ANSI standard has a Level M (for Minimal), which is below Level 0A, so NCAR’s current GKS package requires a higher level of functionality than the minimal level defined by ANSI.
The NCAR-developed higher-level FORTRAN graphics utilities (CONREC, CONRAN, VELVCT, etc.) have been rewritten in FORTRAN 77 and converted to rely on GKS Level 0A in place of the NCAR System Plot Package. The goal for the initial package release was to duplicate the functionality of the old package within the GKS environment. Some changes and improvements have been made. The EZMAP package has been substantially rewritten to add new capabilities and correct old errors. A new utility called HSTGRM has been produced for creating histograms. The System Plot Package entry GRIDAL (for producing various backgrounds) has been rewritten and promoted to the status of a higher-level utility.

This release is the first of NCAR's GKS package, and as such, its users should anticipate some errors in the package. You are encouraged to report errors and make comments on the code or documentation. The official NCAR contact for such feedback is Stu Henderson at (303) 497-1295. Errors will be corrected and a revised release will be made as soon as possible; note, however, that the speed with which this work is accomplished will depend upon the number of error reports received and the number of available NCAR staff to work on them. This package has high priority in the Graphics Project work queue (especially the NCAR higher-level utilities), but no guarantees of support can be made.

Package Contents

The current documentation consists of:

1. An Overview of NCAR GKS-compatible Graphics Package
2. The Release Contents
3. A Guide for Converting from the NCAR System Plot Package to GKS
4. The NCAR FORTRAN 77 GKS Graphics Utilities
5. SPPS: An NCAR System Plot Package Simulator Guide
6. The Elements and Format of the NCAR CGM Metafile
7. The NCAR Computer Graphics Metafile Translator

The code currently available for distribution is written in FORTRAN 77. The document entitled Release Contents details all files currently available and provides a brief description of each file.

Briefly, the code is divided in the following manner:

1. A GKS 0A Package. The version distributed corresponds to GKS Version 7.4. The NCAR code allows for only one active workstation: a Metafile Output (MO) workstation. The format of the created metafile adheres to the emerging Computer Graphics Metafile (CGM) standard. For details on implementing the GKS package, see the section on implementation later in this overview, and for details on the format and contents of the NCAR CGM metafile, see the document Elements and Format of the NCAR CGM Metafile.

2. 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 carefully read the SPPS document for further details on SPPS.
3. **Higher-level utilities.** These are the old NCAR utilities for contouring, world map projections, automatic graphing, 3-D line drawing, and so on, which have been converted to FORTRAN 77 and GKS. A new utility called HSTGRM has been written for producing histograms. GRIDAL (automatic grid-producing package) is also now part of this family. For details, consult the *Utilities* document.

4. **User aids.** The routine FINDG will aid in converting old program units to the new package because it will locate all occurrences of calls to old utilities and calls to System Plot Package entries.

5. **Support files.** These include a FORTRAN 77 version of the PORT error package (ERPRT77), and machine-specific parameters (IIMACH and R1MACH).

6. **Demo drivers.** A simple test is driver supplied for each higher-level utility.

7. **Metafile translator.** A table-driven metafile translator has been written. For details on this translator, see the “Usage Overview” section of this document, and consult *The NCAR Computer Graphics Metafile Translator*. The metafile generated by the NCAR GKS 0A package corresponds to version 7.4 of the emerging ANSI/ISO CGM (Computer Graphics Metafile) standard.

**Implementation Guide**

A logical sequence for implementing the package is as follows:

1. GKS 0A
2. SPPS
3. Metafile translator
4. Utilities/Demo drivers (as needed).

If a GKS package has already been implemented, then steps 1 and 3 above can be avoided.

Now, consider each of these items separately:

**GKS 0A**

In order to implement this package (as well as SPPS), you must write several machine-dependent support routines. Functional descriptions for these routines are in Appendix B of *A Guide for Converting Program Units from the NCAR System Plot Package to GKS*. IAND, IOR, and ISHIFT should be easy to implement. The parameters needed for IIMACH and R1MACH for all standard configurations are in the distributed files IIMACH and R1MACH. Just locate the parameters desired and remove the comment flags from them. A sample G01MIO is distributed as part of the BWI file (see below). This version of G01MIO associates the FORTRAN logical output unit with file name GMETA for the metafile output name. You can modify this subroutine if you want some other output file for the metacode.

The primary work required of the implementor will be the implementation of GBYTES and SBYTES. FORTRAN implementations of GBYTES and SBYTES are distributed as part of the support routines file. These implementations are portable (they depend only upon IIMACH), but implementors should be cautioned that these portable implementations run inefficiently, and more efficient versions should be supplied. For those implementors having access to a “C” compiler, “C” implementations of GBYTES and SBYTES are provided.
A test driver for all of the required support routines is distributed. This driver can serve as a verification tool for the implementations of the support routines.

The GKS code is split into two parts on the distribution tape. The files are divided into AWI (for Above Workstation Interface), and BWI (for Below Workstation Interface). The code in AWI gathers the parameters from the various GKS calls into a single common block and then, when appropriate, invokes the code in BWI to generate the appropriate metafile instructions. In theory, it would be possible to replace BWI with a personally coded device driver for a given graphics output device. This has not yet been attempted, and it is unknown how much work would be involved.

**SPPS**

The System Plot Package Simulator must be implemented before implementation of any of the higher-level utilities. SPPS relies only on GKS Level 0A calls, so implementation should be a matter of compilation with any GKS 0A package. SPPS does invoke GKS 0A functions which are not a part of the ANSI GKS Level MA. Also, note that SPPS does not contain an emulation of the old FLASH package. The functionality of the old FLASH package is available in a GKS Level 2A or higher implementation. The old System Plot Package contained an entry FLUSH. This entry name may conflict with a system-level entry of the same name on many machines. FLUSH is included only to support the old SPP entry, and it is never called anywhere in the current package. To avoid conflicts with a system subroutine of the same name, you can eliminate the FLUSH entry of SPPS, or rename it if current applications require it.

**Metafile translator**

When executing the NCAR GKS package, you obtain a metafile that adheres to the emerging ANSI/ISO CGM (Computer Graphics Metafile) standard. In order to produce plots from any of the higher-level utilities, you must translate the metafile produced for the graphics device(s) used. In order to implement the NCAR CGM translator (file GTRANS on the distribution tape), you must implement some machine-specific routines. Functional descriptions for these routines are in section IV of *The NCAR Computer Graphics Metafile Translator*. For those who have access to a “C” compiler, “C” implementations of the machine-specific support routines are distributed as part of the support files.

To configure itself for a specific graphics output device, GTRANS reads in a configuration file for the device in question and initializes itself accordingly. The configuration files for graphics output devices are called GRAPHCAP files. A list of publically supported and distributed GRAPHCAP files is contained in *The NCAR Computer Graphics Metafile Translator*. If a GRAPHCAP exists, all you need to do is make the correct file connection for GTRANS to read the GRAPHCAP file; if a GRAPHCAP file does not exists, then you must create one in the manner described in Section II of *The NCAR Computer Graphics Metafile Translator*. The GRAPHCAP files that are read by the translator are binary files. To create these binary files, you need to process the character-table form of the GRAPHCAP file by executing the program GRAPHC to read in the character form of the GRAPHCAP. This procedure will produce the desired binary file.

In addition to GRAPHCAP files, the translator reads in a FONTCAP file to describe the available character fonts. FONTCAPS are handled in a manner completely analogous to GRAPHCAP files.
To obtain the character-table for a publically supported GRAPHCAP or FONTCAP, locate the desired table in the distributed files GRAPHCAPS and FONTCAPS and copy only that part to a file. Then, compile and load the GRAPHIC and FONTC files. After you execute the module obtained from GRAPHIC, you will be asked not only for the name of the character-table file you just pulled out of GRAPHCAPS, but also for the name of the output file to receive the binary GRAPHICAP. Binary fontcap files can be similarly created from the character-table FONTCAP.

You can find a list of publically supported and distributed GRAPHCAP files in the document The NCAR Computer Graphics Metafile Translator.

Execution of the metafile translator (file GTRANS and the necessary support routines) requires that the translator have access to the files GRAPHICAP, FONTCAP, and the input metafile.

This section describes how the translator opens the GRAPHICAP and FONTCAP files. Within the translator, the graphcap file name is stored in variable GRCAPN (GRCAPN is a CHARACTER*1 (80) variable). GRCAPN has a default value of "GRAPHICAP", and the call

```
CALL BOPRED(UNIT,GRCAPN,IOS,STATUS)
```

is executed to open the GRAPHICAP file. The FONTCAP file name is stored in the CHARACTER*1 (80) variable FNCAPN; the default value of FNCAPN is "FONTCAP". On UNIX systems, you can set environment variables GRAPHICAP and FONTCAP and you can use "getenv" in BOPRED to open the files; on other systems, changes may be required in the translator code. See Section IV of the translator document for complete details on opening the GRAPHICAP and FONTCAP files.

To connect to the actual metafile, you may configure the translator in one of two ways. First, in the default case, you can use the locally-implemented support routines ARGGET and IARGCT to obtain the metafile name from the command line and to perform the necessary OPEN. Second, you can have the translator read the metafile name from the standard input. In this case, you must set the LOGICAL variable METIPT to .TRUE. in the BLOCKDATA TRNDAT subroutine, and also supply dummy routines ARGGET and IARGCT.

For complete details on the NCAR CGM translator, see The NCAR Computer Graphics Metafile Translator.

**Higher-level utilities**

Once SPPS has been implemented, the higher-level utilities can be implemented. The implementation of these utilities should be straightforward; any special implementation instructions should be contained in the documentation for the appropriate file in the Utilities document. Currently, all utilities rely only on GKS Level 0A subroutine calls. If you want to convert previously existing programs to GKS use, see A Guide for Converting from the NCAR System Plot Package to GKS.

To invoke the higher-level utilities, check the appropriate documentation in the Utilities document. Some differences do exist between calling sequences of the old NCAR utilities that relied on the NCAR Plot Package and the new utilities in the current package. These differences are documented in A Guide for Converting Program Units from the NCAR System Plot Package to GKS. You are encouraged to read the SPPS document carefully, since there are differences
between use of the SPPS entries in the current GKS-based package and use of analogous entries in the old NCAR System Plot Package.

Before invoking any of the higher-level utilities, GKS must be opened, and a workstation opened and activated.

To open GKS, issue the call:

\begin{verbatim}
CALL GOPKS (6,ISZ)
\end{verbatim}

This call specifies unit 6 as the logical unit to receive GKS error messages. ISZ is the maximum buffer size to be allowed for GKS--this is a dummy argument as far as NCAR's GKS 0A package is concerned.

To open a workstation, issue the call:

\begin{verbatim}
CALL GOPWK (1, 2, 1)
\end{verbatim}

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 MO workstation in the NCAR GKS 0A package). With the routine G01MIO provided, the invocation GOPWK will open FORTRAN logical unit 2 with file name GMETA.

Finally, to activate the workstation, issue the call:

\begin{verbatim}
CALL GACWK (1)
\end{verbatim}

When plotting is completed, the workstation must be deactivated and closed, and GKS must be closed. This procedure can be accomplished with the following calls:

\begin{verbatim}
CALL GDAWK (1) 
CALL GCLWK (1) 
CALL GCLKS
\end{verbatim}

Single calls to perform the necessary opening and closing described above are provided in the SPPS package and are described in the SPPS document.

In their current form, the higher-level utilities are meant to duplicate the functionality of the old NCAR System Plot Package. In the default case, the current attribute settings of GKS are saved by invoking a higher-level utility, and restored by returning from that utility. The default line color and text color is the foreground color. A crude interface has been provided to change major and minor line colors and text color in the output from the utilities. AUTOGRAPH and EZMAP have their own color access interface, so you should consult the documentation to learn the interface for those individual packages. For all other utilities, a common block has been added that contains the color indices to be used for major lines, minor lines, and text. All three of the color indices are initialized to "1", the foreground color. If you want to change colors, then change the definition of color index 1 (by using the GKS function GSCR), or 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.

Please note that this color scheme has not been well-tested.
The following is a table of the common block in each utility, and the names of the three color indices in that utility.

<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>conrcqck</td>
<td>qckint</td>
<td>iqckmj</td>
<td>iqckmn</td>
<td>iqcktx</td>
</tr>
<tr>
<td>conrespr</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>idchmj</td>
<td>idchmn</td>
<td>idchtx</td>
</tr>
<tr>
<td>dashline</td>
<td>dlnint</td>
<td>idlnmj</td>
<td>idlnmn</td>
<td>idlnxt</td>
</tr>
<tr>
<td>dashsmth</td>
<td>dsmint</td>
<td>idsmmj</td>
<td>idsmmn</td>
<td>idsmtx</td>
</tr>
<tr>
<td>dashsupr</td>
<td>dpint</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>igritx</td>
</tr>
<tr>
<td>hafton</td>
<td>hafint</td>
<td>ihafmj</td>
<td>ihafmn</td>
<td>ihaftx</td>
</tr>
<tr>
<td>isosrf</td>
<td>isoint</td>
<td>iiisomj</td>
<td>iiisomn</td>
<td>iisotx</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>ipwtxt</td>
</tr>
<tr>
<td>pwrity</td>
<td>pwyint</td>
<td>ipwymj</td>
<td>ipwymn</td>
<td>ipwytx</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>ipzttx</td>
</tr>
<tr>
<td>srface</td>
<td>sr fint</td>
<td>isrfinj</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>ithrtx</td>
</tr>
<tr>
<td>velvct</td>
<td>velint</td>
<td>ivelmj</td>
<td>ivelmn</td>
<td>iveltx</td>
</tr>
</tbody>
</table>
Release Contents

The National Center for Atmospheric Research
Scientific Computing Division

*NCAR is sponsored by the National Science Foundation*
Release Contents
NCAR GKS-Based Graphics Software

The list below separates the NCAR GKS-based Graphics package into logical components and discusses the files that comprise each component. Mixed-case bold-face titles running into the text define the logical components. Individual files, as they will be known on the tape, are identified by the upper-case bold-face titles in the left margin. Routines of key interest in some of the files are listed in upper-case bold-face and run into the text.

Unless otherwise noted, assume that all software in the NCAR GKS-based Graphics package is written in FORTRAN 77 and adheres strictly to that standard.

Included in the description of a file is a list of other files needed to properly assemble and run that file. However, the routines in the files LOCAL, SUPPORT, and SPFS are used throughout the package, and thus it is assumed that they will be available to every routine.

NCAR GKS-0A

**AWI** Standing for “Above Workstation Interface,” AWI codes up the parameters from the various GKS calls into a single common block and then, when appropriate invokes BWI (see below).

**BWI** Standing for “Below Workstation Interface,” upon being invoked by AWI it takes the coded up parameters in a single common block, and generates the appropriate CGM metafile instructions.

Required Support Routines

**SUPPORT** This file contains a collection of portable FORTRAN 77 routines required by code throughout the package. It should be implemented in all cases.

**ENCOD** Used by the VELVCT and CONREC Utilities to generate ASCII labels from numeric values by encoding them in an appropriate format.

**ERPRT77** An error handling package, adapted from the non-proprietary part of the PORT Mathematical Subroutine Library from Bell Labs.

**Q8QST4** A “stub” or “dummy” routine called from all Utilities to substitute for the accounting subroutine used at NCAR.

**LOCAL** This file contains the examples of the lowest level machine-dependent (local) subroutines that need to be implemented on each machine. FORTRAN and/or C versions of these routines are provided as samples of what should be done, but it is recommended that they be implemented in assembly language since they are used extensively by the entire package.

**IOR** A function which results in the bit-by-bit logical sum of its two arguments.
The NCAR Graphics Package

IAND: A function which results in the bit-by-bit logical product of its two arguments.

ISHIFT: A function which, based on the integer value of its second argument, N, will perform on its first argument a left circular shift by N bits if N is positive, or right end-off shift by N bits if N is negative.

GBYTES: Unpacks “bytes” from a bit-string given in an array, and puts them into another array, with one “byte” per array element. A “byte” is here defined not as a string of 8 bits, but rather as an atomic or morphemic item of information which may be one or more bits.

SBYTES: Packs “bytes” from an array with one “byte” per element into a bit-string given in another array. A “byte” is here defined not as a string of 8 bits, but rather as an atomic or morphemic item of information which may be one or more bits.

IIMACH: A function which returns in an array, 16 machine-dependent integer-valued constants. These constants are given for a large number of machines within the function as commented lines; simply remove comments from the appropriate section. This is the NCAR version of a nonproprietary part of the PORT Mathematical Subroutine Library from Bell Labs.

R1MACH: A function which returns in an array, 5 machine-dependent real-valued constants. These constants are given for a large number of machines within the function as commented lines; simply remove comments from the appropriate section. This is the NCAR version of a nonproprietary part of the PORT Mathematical Subroutine Library from Bell Labs.

Implementation Tools

IMPLTEST TLOCAL: A program which tests correctness of implementation for the required local routines IAND, IOR, ISHIFT, GBYTES, and SBYTES.

TGKSOA: A program which generates an example of the five GKS output primitives.

TSPPS: A program which extensively tests all SPPS (System Plot Package Simulator) entries.

TPLOT: A program which produces a single simple plot which depends only on GKS and SPPS. The code contains octal and hex dumps of the metafile which should be produced.

FINDG A program for locating all calls to NCAR System Plot Package entries. This program is useful for converting from the old System Plot Package system to GKS.

Release Contents 2 July 1986
The NCAR Graphics Package

Utilities

AUTOGRPH  Draws and annotates curves or families of curves. The standard version of this package requires DASHCHAR. The smoothed version requires DASHSMTH.

CONRAN   Contours irregularly spaced data, labelling the contour lines. The standard version of this package requires DASHCHAR. The smoothed version consists of this file with DASHSMTH. Also requires CONCOM and CONTERP.

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

CONRAQ   Like CONRAN, but smaller and faster because it has no labelling capacity. Also requires CONTERP.

CONREC   Contours two-dimensional arrays, labelling the contour lines. The standard version of this package requires DASHCHAR. The smoothed version, CONRECSMTH, consists of this file with DASHSMTH.

CONRCQCK  Like CONREC standard, but faster and smaller because contours are unlabelled. This package shares entry names with CONREC, so they cannot both be included in a binary library.

CONRCSPR  Like CONREC, but bigger and slower because contours are smoothed, labelled, and crowded lines are removed. This package shares entry names with CONREC, so they cannot both be included in a binary library.

DASHCHAR   Software dashed line package with labelling capability.

DASHLINE   Like DASHCHAR, but smaller and faster because it has no labelling 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 cannot 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 and/or United States state outlines according to one of nine projections. Requires the continental outline data base EZMAPDAT.

GRIDAL   Package for drawing graph paper, backgrounds, perimeters, etc.
The NCAR Graphics Package

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

HSTGRM  General purpose package for drawing histograms.

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

ISOSRFHR Iso-valued 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.

PWRY     Draws simplest software characters.

SRFACE   Three-dimensional display of a surface (with hidden lines removed) from a two-dimensional array. May optionally include PWRZS, for drawing characters in three space.

STRMLN   Plots a representation of a vector field flow of any 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. May optionally include PWRZT, for drawing characters in three space.

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

Utility Support Files

SPPS     Standing for the "System Plot Package Simulator," this collection of routines simulates the old NCAR System Plot Package by containing the same entry points which in turn call the appropriate GKS routines to accomplish the given task. These routines are used extensively by the Utilities above. For users of the old package, these routines should prove helpful in the conversion to the new GKS-based package. New users should find some of the entry points convenient; for example, SPPS provides logarithmic scaling of plots, whereas the GKS standard does not.

CONCOM   A set of routines used by the CONRAN and CONRAS packages.

Release Contents 4 July 1986
CONTERP  A set of routines used by the CONRAN, CONRAQ, and CONRAS packages.

PWRZI  May optionally be included with the ISOSRF Utility to draw characters in three space.

PWRZS  May optionally be included with the SRFACE Utility to draw characters in three space.

PWRZT  May optionally be included with the THREED Utility to draw characters in three space.

BNCHMK  A program which generates six non-trivial examples of usage of the Utilities, particularly examples of overlaying the output from one Utility onto another. It calls the following Utilities: GRIDAL, AUTOGRPH, CONREC, DASHCHAR, VELVCT, and EZMAP. Additionally, it requires the data base file RANFDAT, which is a separate file on the tape.

UTILTEST  The graphics manual that is provided with this distribution contains sample plots from each of the Utilities. The routines that produced these demonstration plots are included in this file, and can be used to test-run any Utility and compare its output with the output provided in the manual. A driver program that calls one or more of these Utilities is not provided, but must be written locally.

TAUTOG: Demonstration subroutine for the AUTOGRPH Utility.

TCNQCK: Demonstration subroutine for the CONRCQCK Utility.

TCNSMT: Demonstration subroutine for the CNRCSMTH Utility.

TCNSUP: Demonstration subroutine for the CONRCSUPR Utility.

TCONAN: Demonstration subroutine for the CONRAN Utility.

TCONAQ: Demonstration subroutine for the CONRAQ Utility.

TCONAS: Demonstration subroutine for the CONRAS Utility.

TCONRE: Demonstration subroutine for the CONREC Utility.

TDASHC: Demonstration subroutine for the DASHCHAR Utility.

TDASHL: Demonstration subroutine for the DASHLINE Utility.

TDASHP: Demonstration subroutine for the DASHSUPR Utility.

TDASHS: Demonstration subroutine for the DASHSMTH Utility.

TEZMAP: Demonstration subroutine for the EZMAP Utility.

TGRIDA: Demonstration subroutine for the GRIDAL Utility.

THAFTO: Demonstration subroutine for the HAFTON Utility.

THSTGR: Demonstration subroutine for the HSTGRM Utility.

TISOHR: Demonstration subroutine for the ISOSRFHR Utility.

TPWRTX: Demonstration subroutine for the PWRITX Utility.
TPWRY: Demonstration subroutine for the PWRY Utility.
TPWRZI: Demonstration subroutine for the PWRZI Utility.
TPWRZS: Demonstration subroutine for the PWRZS Utility.
TPWRZT: Demonstration subroutine for the PWRZT Utility.
TSRFAC: Demonstration subroutine for the SRFACE Utility.
TSTRML: Demonstration subroutine for the STRMLN Utility.
TTHREE: Demonstration subroutine for the THREED Utility.
TVELVC: Demonstration subroutine for the VELVCT Utility.

CGM Translator

CGMTRANS

A Program which reads the metafile and translates it into plot instructions for a given graphics device as defined by the appropriate GRAPHCAP file. A collection of GRAPHCAPs is provided (see Data Bases below). To run this program the GRAPHCAPs must first be converted from their distributed ASCII format into binary format with the program GRAPHC, and the FONTC program must be run on the FONTCAP file to produce a similar binary font file. Additionally, the CGM Translator support routines must be implemented (see below).

TRNSPPRT

This file contains examples of the machine-dependent (local) subroutines which need to be implemented on each machine for either the CGM Translator or for the GRAPHC or FONTC programs. FORTRAN and/or C versions of these routines are provided as samples.

ARGGET: Get the requested argument from the command line.
BCLRED: Close a file opened by BOPRED for sequential binary reading.
BINCLS: Close a file opened for sequential binary writing. The file was opened by BINOPN.
BINOPN: Open a file for sequential binary writing.
BINRED: Transfer "COUNT" FORTRAN integers from an unformatted file which has been opened by BOPRED.
BINWRI: Transfer the contents of a buffer to the named file. Use a sequential unformatted write to perform the transfer. The file was opened by BINOPN.
BOPRED: Open a file for sequential unformatted reading. The file was created either by the GRAPHCAP or FONTCAP preprocessors using the local routine BINOPN.
CHRCLS: Close a file opened for sequential character reading. The file was opened by CHROPN.
CHROPN: Open a file for sequential unformatted reads. The file contains characters with a maximum of 80 characters per record.
The NCAR Graphics Package

CHRRED: Transfer from a character file to a FORTRAN character array. The file was opened by CHROPN.

FLUS: Dump device-dependent instructions.

FRPRMP: Send the "frame finished" prompt and wait for response (used only if the device is declared NON-BATCH).

IARGCT: Used only when optional command line processing is done, this routine returns as the function result the number of arguments on the command line.

MSSG: Put out a message on the error unit which corresponds in format to the Status Message section in the Graphics manual.

MTOPEN: Open a file for direct unformatted reading of 1440 byte records. The file is a metafile generated by BWI (see GKS OA above).

READIT: Read a character string from standard input and put it into a FORTRAN 77 CHARACTER*1 array.

RECRED: Read in a metafile record using direct access I/O. The metafile is opened by the MTOPEN routine.

WRITIT: Write a character string to standard output.

Data Bases

EZMAPDAT  Continental outlines data base for the EZMAP Utility.

PWRITXC1  Part I of card image representation of PWRITX Complex font digitization. Must be run through the PWRITXNT preprocessor to convert it to a binary form which the PWRITX Utility expects.

PWRITXC2  Part II of card image representation of PWRITX Complex font digitization. Must be run through the PWRITXNT preprocessor to convert it to a binary form which the PWRITX Utility expects.

PWRITXD1  Part I of card image representation of PWRITX Duplex font digitization. Must be run through the PWRITXNT preprocessor to convert it to a binary form which the PWRITX Utility expects.

PWRITXD2  Part II of card image representation of PWRITX Duplex font digitization. Must be run through the PWRITXNT preprocessor to convert it to a binary form which the PWRITX Utility expects.

FONTCAP  Master file of all FONTCAPs for the CGM Translator. In ASCII form, it must be run through the FONTC preprocessor (below) to convert it to a binary form which the CGM Translator expects.
GRAPHCAP  Master file of all GRAPHCAPs for the CGM Translator. In ASCII form, a GRAPHCAP for a specific graphics device must be gleaned from this file and be run through the GRAPHC preprocessor (below) to convert it to a binary form which the CGM Translator expects. Included in this files are GRAPHCAPs for: ADM5 with DEC RG1000 graphics board, AED512 in ASCII mode, AED512 in binary mode, HI DMP-29 in small chart mode, HP150 PC, HP7475 in ASCII mode, HP2648A in binary mode, IMAGEN 8/300 laser printer in graphics landscape mode, IMAGEN 8/300 laser printer in graphics portrait mode, IBM PC (and compatibles) monochrome to drive PCPLOT, RAMTEK 6211 in Tektronix compatible mode, Selinar HiREZ100, Tektronix 4010, 4012, 4025, 4105, 4107, and 4115B, TEKALIKE program on the APPLE MACINTOSH, VERSATERM program on the APPLE MACINTOSH, Digital VT100 with VT640 retrofit, Digital VT125, and a blank form for user oriented graphcap definition.

RANFDAT  Data base for BNCH test routine included in the UTILTEST file.

Data Base Initializers

PWRITXNT  Preprocessor for the PWRITX data base files PWRITXC1, PWRITXC2, PWRITXD1, and PWRITXD2 to convert them into binary PWRITX data base.

FONTC  Preprocessor for the FONTCAP data base file to convert them into a binary form needed by the CGM Translator. Also required are several routines from the TRNSPPRT file.

GRAPHC  Preprocessor for the GRAPHCAP data base file to convert them into a binary form needed by the CGM Translator. Also required are several routines from the TRNSPPRT file.
A Guide for Converting Program Units from the NCAR System Plot Package to GKS

The National Center for Atmospheric Research
Scientific Computing Division

NCAR is sponsored by the National Science Foundation
Conversion Guide

Table of Contents

Introduction ........................................ 1
Converting to FORTRAN 77 ......................... 1
Changes to NSPP Calls ................................ 2
Automatic Log-scaling .............................. 2
Eliminating Absolute Integer Addressing ........ 2
Changes to Higher-level Utility Calls ........... 3
Eliminating Outmoded NSPP Routines .......... 4
Appendix A: FORTRAN 77 CHARACTER Variables 6
Appendix B: Machine-dependent Support Routines 11
Appendix C: PORT Error Handling Package .... 15
Conversion Guide

Introduction

This document is a guide to assist you in converting program units from using the FORTRAN 66 NCAR System Plot Package (NSPP) and higher-level FORTRAN utilities to using the new GKS-compatible NCAR Graphics Package routines that have the same functionality. The text includes a step-by-step sequence on how to convert an NSPP-based code to a FORTRAN 77 GKS-based code.

Almost all of the NCAR higher-level FORTRAN graphics utilities have been converted to a FORTRAN 77 package compatible with GKS Level 0A. The exception is SCROLL, which uses the FLASH logic from the old NCAR System Plot Package. The FLASH logic capability is not included in the NCAR Level 0A version (see below). The WINDOW package will not be a part of the GKS package. The function performed by WINDOW is known as clipping — cutting away all parts of a picture which lie outside of the user-defined window. Since clipping is a part of GKS functionality, the WINDOW package is unnecessary. All of the demonstration drivers for all utilities have been converted to FORTRAN 77 and GKS.

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, then emulation of the FLASH package is a straightforward matter.

Because eliminating all NSPP calls in favor of GKS calls would be a difficult task — especially since the functionality of GKS 0A and the NSPP is not identical — a package of GKS-based subroutines that emulates the NSPP has been written. You should familiarize yourself with this package. For further details, see SPPS — An NCAR System Plot Package Simulator.

To convert a given program unit that uses NSPP to the new GKS package you must perform the following tasks:

1. Convert to FORTRAN 77.
2. Make necessary changes to the old NSPP calls.
3. Implement the new log-scaling scheme.
4. Eliminate the absolute integer addressing allowed in the NSPP but not allowed in GKS.
5. Make appropriate changes in calls to higher-level utilities.
6. Eliminate outmoded support routines that were called directly by the user.

Following is a detailed discussion of each of the preceding items.

Converting to FORTRAN 77

Converting to FORTRAN 77 is simply a matter of performing the language conversion from FORTRAN 66. However, you may have difficulty converting to the use of FORTRAN 77 type CHARACTER variables. For notes on FORTRAN 77 type CHARACTER variables, see Appendix A.
Making Necessary Changes to the NSPP Calls

The most commonly-required change you must make is elimination of the integer address coordinates. To change the NSPP calls, you must first locate the calls in question, which may be a problem. To assist you, there is a portable FORTRAN utility called FINDG. FINDG reads the FORTRAN utility you wish to convert as input from logical unit 5, and writes its output to logical unit 6. Except in the case where the input file contains a line which begins with a "$", the output from FINDG contains a list of line numbers in the input file where a call to an NSPP entry occurred, as well as a copy of the line in the input file that contains the call. To obtain a list of all the subroutine names for which the utility is searching, place a "$" as the first character of any line, and such a list will be displayed. Note that the utility searches only for calls; comment cards that contain one of the search names will not appear in the output list.

Once you have located the appropriate calls, proceed as advised in the SPPS document.

Implementing an Automatic Log-scaling Scheme

GKS does not provide facilities for automatic log-scaling. However, since 20% of all graphics jobs that run at NCAR use a non-default value for the log-scaling parameter of the SET call, automatic log scaling is desirable.

An automatic log-scaling scheme has been implemented for use with the new GKS-based Graphics Utilities. The simplest 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, GTX, etc. The final argument to SET governs only what is produced by the NCAR GKS-based utilities. If you desire log-scaling when using GPL, you will have to do the scaling yourself before calling GPL.

Eliminating Absolute Integer Addressing

Absolute integer addressing was allowed in the old NSPP, but is not allowed in the new GKS-based version. In the NSPP, plotter address units (PAU's) are integer addresses (in the range 1 to 1024 by default) and are used in two ways:

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

PAUs used for absolute addressing, as in item 1 above, have been eliminated from the GKS package. 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 item 1, any such addressing in the GKS utilities must be implemented in the software by using some mechanism such as INTT (a function to determine if an argument is an integer or a floating-point value). However, INTT is impossible to implement on some machines. 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 it is believed that this would introduce more confusion than the reward would warrant.
PAUs are used extensively throughout the old NSPP as absolute lengths, as in item 2. For example, they are used as arguments to PWRIT, PWRTX, 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.

Some comments are needed here with regard to selecting the proper size of lengths as defined in item 2. The length of a PAU in the 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. Two new parameters called “XF” and “YP” have been added to the list of user-settable utility state variables in the subroutines SETUSV and GETUSV (see the SPPS documentation for more details). The purpose of these parameters is identical to that of the parameters in the old SETI subroutine. So when determining the length of a line, or the height of a character specified in PAUs, you will need to check the values of “XF” and “YP” to correctly compute the fraction of the screen size.

**Making Appropriate Changes to Calls to Higher-level Utilities**

Calls to higher level utilities have changed in two significant ways:

1. The coordinate addressing in the new package is strictly in world coordinates;

2. All character manipulations in the new package are handled by FORTRAN 77 TYPE CHARACTER variables.

First, regarding the coordinate addressing, you should note that the alternate absolute coordinate addressing allowed in the NSPP is no longer available. This change results from the conversion to FORTRAN 77, because in FORTRAN 77, a variable of type REAL cannot receive an integer value; therefore, the integer/floating-point specification of coordinates allowed in the NSPP had to be removed. To ease in 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 can now be replaced with the appropriate function applied to the old coordinates. Consult the SPPS document for further details.

The second significant change to the package is that FORTRAN 77 TYPE CHARACTER variables are now used to perform character manipulations. All character information passed to the utilities should now be accomplished by using quoted strings, or CHARACTER variables, and not by using integer or floating-point variables.

Examine, for example, the effect of coordinate and character information changes in the subroutine PWRTX. The first two arguments to PWRTX 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 PWRTX 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.
The option-setting routines of the CONRAN, CONRAQ, and CONRAS packages have been changed to make them conform to FORTRAN 77. All calls to CONOPT should be examined carefully for correctness. Remember that by using the supplied utility, FINDG, you can locate all calls to utilities and entries to the NSPP that may have to be changed.

A primary goal in the conversion of the higher-level utilities to FORTRAN 77 is to retain as much similarity as possible between the argument lists in the new GKS-based package and the argument lists in the NSPP-based package. One major change which had to be made was in the arguments to DASIID, which allowed specification of dash patterns as either character or integer values. To preserve this functionality in the new package, 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.

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)
```
in which the arguments are identical to those for the SPPS routine WTSTR. For details on WTSTR, see SPPS documentation.

### Eliminating Outmoded Support Routines in the NSPP

This task is necessary only if you directly call any of the 14 locally-implemented support routines for the NSPP. Nine of the 14 locally-implemented support routines for the NSPP are being deleted because of various portability problems or other problems. These routines are:

- ENCODE
- GETCHR
- INTT
- LOC
- PACKUM
- PERROR
- SETCHR
- ULIBER
- WRITEB

The new GKS package will have 8 locally-implemented support routines:

- GBYTES
- G01MIO
- IAND
- IOR
- ISHIFT
- I1MACH
- R1MACH
- SBYTES
Note that IAND, ISHIFT, IOR, I1MACH, and R1MACH were also support routines in the NSPP. For a detailed description of the 8 machine-dependent GKS support routines, see Appendix B. Please note that these support routines are for the GKS OA package and for the higher-level utilities. Support routines required to implement the new CGM translator are discussed in The NCAR Computer Graphics Metafile Translator.

Deleting the nine routines listed above results in several changes. For instance, ULIBER and PERROR are replaced by the new centralized error-handling routines. The PORT error handling routines are used for all converted utilities. All ULIBER and PERROR calls should be replaced with the appropriate calls to the PORT routines. Because the official PORT routines are written in FORTRAN 66, they have been rewritten in FORTRAN 77 in the package ERPRT77. Also, since the subroutines have been altered, they have been renamed by removing the final character of each routine name. Thus, the new routine names are SETER, ENTSR, RETSR, NERRO, ERROF, EPRIN, and FDUM.

The calling sequence to SETER is

```
CALL SETER(MESSG,NERR,IOPT)
```

in which MESSG is a CHARACTER variable. Note that it is no longer necessary to specify the number of characters in the message; in addition, the maximum length of the message MESSG is now 113 characters, instead of 72 characters, as previously. The ERPRT77 package will automatically write "ERROR nnnn IN" as the first part of every error message, so begin all messages with the program unit name in which the error occurred. All errors are written to I1MACH(4) — the default error file. For a complete description on how to use the PORT error routines, see Appendix C.

You should also be apprised of several other changes that occur after eliminating NSPP's outmoded support routines.

- The routines WRITEB and PACKUM should not be called directly by any of the old higher-level utilities, so deleting them will not present a problem. The ENCODE routine can be replaced with a statement that performs a write to an internal file, as provided for in FORTRAN 77.

- You must replace GETCHR and SETCHR with the appropriate manipulations on type character variables. This process may require some understanding of the code in question, but in general the replacement should be fairly straightforward.

- Because the LOC function is impossible to implement on certain machines, it has been deleted. However, this deletion may require significant coding changes and perhaps new restrictions. To compensate for the loss of LOC, the person doing the conversion must decide how best to live without it.

Note that the newly converted routines will only accept floating-point world coordinates. This restriction was not mandatory in the NSPP. In most cases, it will eliminate the need for INTT. For a more detailed description of absolute integer coordinate addressing and Plotter Address Units, see the SPPS document.
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 \text{I} in the subroutine \text{SUB} assumes the internal character representation of the string “AB”.

Because different machines 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 ad hoc and non-portable manner. Thus, the TYPE CHARACTER statement has been added to the FORTRAN standard.

The 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 (see Appendix C of the FORTRAN 77 Standard), although many compilers accept it as a non-standard extension to the FORTRAN 77 standard. For example, CFT, the CRAY FORTRAN compiler, accepts the “nH” specifier on CRAY computers.

Before illustrating how to convert FORTRAN 66 Hollerith data items to FORTRAN 77 constructs, it is important to examine the FORTRAN 77 TYPE CHARACTER statement. For these details, see the full definition of the FORTRAN 77 TYPE CHARACTER statement provided in the FORTRAN 77 standard, Section 8.4.2, pg 8-5.

To illustrate the use of the FORTRAN 77 TYPE CHARACTER statement, consider the following examples.

In the simplest instance, one might have the statement

\[
\text{CHARACTER*3 A}
\]

that flags “A” as a type character variable. The compiler reserves enough space on the given machine 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

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

produces

A=ABC

on any machine running a FORTRAN 77 compiler. If you consider 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 machines using FORTRAN 66 without dimensioning the variable A (and typing "A" INTEGER as well) since the machine may only allow 2 characters per word. The standards are intended to eliminate the necessity to perform different operations when handling the same situation on different machines.

Now consider another example of TYPE CHARACTER statements. The statement

```fortran
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

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

produces

I=ABCDE12345

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)
```

Consider yet another example.

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

would 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 or 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
```

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”.

Some intrinsic functions useful in dealing with character variables are described in Table 5 of the FORTRAN 77 standard (pp 15-22 and 15-23). In brief, they are LEN, ICHAR, and CHAR. 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
```

produces

```
J=5, K=65, Y=A
```
on the standard output.

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 when in this predicament. 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
500 FORMAT(' LENGTH = ',I3)
END
```

produces

```
LENGTH = 3
LENGTH = 5
```
on the standard output.

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

```
CHARACTER*3 A
CHARACTER*2 B
CHARACTER*6 C
A = 'ABCD'
B = 'ST'
C = A // B
WRITE(6,500) C
```

produces

```
C = ABCDST
```

on the standard output.
Following are functional descriptions of the required locally-implemented support routines. A test suite is distributed for this package so that an implementor may verify that his implementations are correct. The routine G01MIO is needed only if files AWI and BWI (NCAR's GKS package) are being implemented.

FUNCTION I1MACH(I)

This function is used to set up 16 machine constants:

I1MACH(1) = the standard input unit
I1MACH(2) = the standard output unit
I1MACH(3) = the standard punch unit
I1MACH(4) = the standard error message unit
I1MACH(5) = the number of bits per integer storage unit
I1MACH(6) = the number of characters per integer storage unit

Assume that integers are represented in the S-digit, base-A form:

\[ \text{SIGN}(X(S-1)A^{S-1}+\ldots+X(1)A+X(0)) \]

in which 0.LT.X(I).LT.A for I=0,...,S-1.

I1MACH(7) = A, the base
I1MACH(8) = S, the number of base-A digits
I1MACH(9) = A**S-1, the largest magnitude

Assume that floating-point numbers are represented in the T-digit, base-B form:

\[ \text{SIGN}(B**E)((X(1)/B+\ldots+(X(T)/B**T)) \]

in which 0.LT.X(1), and EMIN.LE.E.LE.EMAX.

I1MACH(10) = B, the base

**Single-precision Constants**

I1MACH(11) = T, the number of base-B digits
I1MACH(12) = EMIN, the smallest exponent E
I1MACH(13) = EMAX, the largest exponent E

**Double-precision Constants**

I1MACH(14) = T, the number of base-B digits
I1MACH(15) = EMIN, the smallest exponent E
I1MACH(16) = EMAX, the largest exponent E
FUNCTION RIMACH(I)

This function sets 5 single-precision machine constants:

\[
\begin{align*}
R\text{IMACH}(1) &= B^{**}(E\text{MIN}-1), \text{the smallest positive magnitude} \\
R\text{IMACH}(2) &= B^{**}E\text{MAX}*(1-B^{**(-T)}), \text{the largest magnitude} \\
R\text{IMACH}(3) &= B^{**(-T)}, \text{the smallest relative spacing} \\
R\text{IMACH}(4) &= B^{**(1-T)}, \text{the largest relative spacing} \\
R\text{IMACH}(5) &= \log_{10}(B)
\end{align*}
\]

FUNCTION ISHIFT(IWORD,N)

IWORD is shifted by N bits. If \( N > 0 \), a left circular shift is performed (all bits are shifted left \( N \) bits, and the bits that are shifted out of the word to the left are shifted back into the word at the right); if \( N < 0 \), a right end-off shift is performed (all bits are shifted right by \( N \) bits, and the bits that are shifted out of the right of the word are lost).

It is required that \( \text{IABS}(N) \leq \text{word length} \).

FUNCTION IAND(K1,K2)

The bit-by-bit logical product of K1 and K2. If \( K3 = \text{IAND}(K1,K2) \), then the \( i \)-th bit of \( K3 \) is 0 if the \( i \)-th bit of either \( K1 \) or \( K2 \) is 0; otherwise the \( i \)-th bit of \( K3 \) is 1.

FUNCTION IOR(K1,K2)

The bit-by-bit logical sum of K1 and K2. If \( K3 = \text{IOR}(K1,K2) \), then the \( i \)-th bit of \( K3 \) is 0 if and only if the \( i \)-th bit of both \( K1 \) and \( K2 \) is 0.

SUBROUTINE G01MIO (IOP, IUNIT, IBUFF, LENGTH, IERROR)

This output routine is the central one for the metafile generator. A FORTRAN implementation of this subroutine is distributed as part of the bwi file. The FORTRAN implementation of G01MIO contained in bwi should work correctly on most machines.

Input Parameters

IOP

Type of operation desired:

\[
\begin{align*}
&= 1, \text{OPEN workstation for output on IABS(IUNIT).} \\
&= 2, \text{CLOSE workstation for output on IABS(IUNIT).} \\
&= 3, \text{write buffer to IABS(IUNIT).}
\end{align*}
\]

IUNIT IABS(IUNIT) is the FORTRAN logical unit number on which IOP is to occur.

IBUFF Buffer containing data for a write operation.

LENGTH Length of data in IBUFF.
Output Parameters

IERROR  Error indicator that equals 0 if no errors.

SUBROUTINE GBYTES(NPACK, ISAM, IBIT, NBITS, NSKIP, ITER)

This subroutine is used to unpack bit chunks from NPACK into the ISAM array. A portable FORTRAN version of this routine is distributed, but the FORTRAN version is inefficient and should be replaced with a more efficient implementation.

NPACK  Address of first word of array to be unpacked.

ISAM  Array to receive the unpacked bit chunks. They will be right-justified with zero-fill in this array. ISAM should be dimensioned for ITER.

IBIT  A bit-count offset to be used before the first bit chunk is unpacked. For example, if IBIT=3, and NBITS=5, then 3 bits in NPACK will be skipped and the next 5 bits will be unpacked into ISAM(1).

NBITS  The number of bits in each bit chunk to be unpacked. An error condition obtains if NBITS is larger than the number of bits per word on the given machine.

NSKIP  The number of bits to skip between each bit chunk to be unpacked. Bits are skipped only after the first bit chunk has been unpacked.
ITER The number of bit chunks to be unpacked. For example:

```
CALL GBYTES(NPB,ISB,3,6,9,2)
```

In this call, 3 bits would be skipped at the beginning of NPB; the next 6 bits would be unpacked into ISB(1) and right-justified with zero-fill; 9 bits would be skipped in NPB, and then the next six bits of NPB would be unpacked into ISB(2) and right-justified with zero-fill.

**SUBROUTINE SBYTES(NPACK,ISAM,IBIT,NBITS,NSKIP,ITER)**

This subroutine is the reverse of GBYTES as described above.

- **NPACK** Address of first word of array to be packed.
- **ISAM** Array to be packed into NPACK. The right-most NBITS bits of each word will be packed. ISAM should be dimensioned for at least ITER.
- **IBIT** A bit-count offset to be used before the first bits are packed into NPACK. For example, if IBIT=3, and NBITS=5, 3 bits in NPACK will be skipped before the right-most 5 bits of ISAM(1) are packed into it.
- **NBITS** The number of bits in each word of ISAM to be unpacked. An error condition obtains if NBITS exceeds the word size on the given machine.
- **NSKIP** The number of bits to skip between each bit chunk packed.
- **ITER** The number of bit chunks to be packed. For example:

```
CALL SBYTES(NPC,ISB,45,6,3,2)
```

In this call, 45 bits would be skipped at the beginning of NPC; the right-most 6 bits of ISB(1) would be packed into NPC; 3 bits would be skipped in NPC, and the right-most 6 bits of ISB(2) would be packed into NPC.
The PORT error package allows for two types of errors: fatal errors and recoverable errors. Generally speaking, fatal errors signal blunders you have made in calling the subprogram—an input parameter, for example, outside of the legal range. Whenever a fatal error occurs, an error message is printed, the run is terminated, and a dump routine is called. A recoverable error, on the other hand, is one you can override. If you do not override the error, the message is printed and the run is terminated as above, but the dump routine is not called.

If you wish to recover from a recoverable error and to gain control over the error-handling process, then use "recovery mode." At any point in a run you can enter recovery mode. While in this mode, you can:

- Determine whether an error has occurred, and, if so determine the error number,
- Print any current error message,
- Turn off the error state,
- Leave the recovery mode.

Seven subprograms dealing with various errors you might encounter are documented in this chapter. They are SETER, ENTSR, RETSR, NERRO, ERROF, EPRIN and FDUM. Their use is described below. Even though the descriptions are given in the context of the higher-level graphics utilities, you are free to directly call on the error-handling mechanism.

When an error is detected in a graphics utility, a call is made to the principal error handling routine, SETER. The calling sequence is

\[
\text{CALL SETER(MESSG,NERR,IOPT)}
\]

in which MESSG is a message of type character, and NERR is the error number. IOPT indicates if the error is a fatal one (IOPT=2) or a recoverable one (IOPT=1). However, unless the recovery mode is in effect, SETER prints an error message and terminates the run.

Whether recovery mode is in effect or not at any given instant is determined by setting an internal recovery switch. To change the setting of the switch, you must call the subprogram ENTSR (enter and set recovery). The call

\[
\text{CALL ENTSR(IROLD,IRNEW)}
\]

sets the recovery switch to IRNEW and returns the previous value in IROLD. If the input value for IRNEW is 1, the recovery mode is entered; if it is 2, the recovery mode is turned off; if it is 0, no change is made. Once the recovery mode has been entered, it is in effect until it is turned off. The recovery mode is off by default in the graphics utilities.

When SETER is called to handle a recoverable error and the recovery mode is on, the error number and the associated message are both stored, and control is returned to the statement following the call to SETER. Thus, if you have entered the recovery mode and called a subprogram that may have had a recoverable error (and may have made a call to SETER...
to store the information on the error), then upon returning from the subprogram you are
responsible for testing the occurrence of an error. The error number can be retrieved and
tested by writing a statement such as the one that follows:

IF (NERRO(NERR) .EQ. 4) GO TO 50

The value of the function, NERRO, and the value of the argument, NERR, are both set to
the current value of the error number by NERRO. Note that if the error number is non-
zero, then an error has occurred and you must correct it.

After correcting the error, turn the error state off by using the statement:

CALL ERROF

If another error occurs before ERROF has been called, the run is terminated because the
occurrence of two errors in a row without an explicit recovery statement between them is
regarded as a fatal error. This difficulty can only result from neglecting to properly recover
from a previous error.

Example

The example below illustrates the use of the subprograms ENTSR, NERRO, and ERROF in
detecting and recovering from an error. The following program fragment calls on a fictitious
subprogram, XMPL(X,EPS,N), that computes a value for X, accurate to within EPS, using
no more than N iterations. XMPL contains one recoverable error, Error 2, which occurs
when the accuracy, EPS, cannot be obtained within N iterations.

The sample program that follows identifies the best answer (smallest order of magnitude for
EPS) that can be obtained by 10 or fewer iterations. After each call to XMPL, the program
calls NERRO to see if Error 2 has occurred, or, in other words, to see whether the current
value of EPS is too stringent. If the Error 2 has occurred, then EPS is enlarged to ten times
its former value, the error is turned off, and XMPL is called again. Note that to be on the
safe side, an upper limit, EPSMAX, is imposed on EPS. Also, remember that the error must
be turned off after the call because the run will terminate if two consecutive errors are
encountered.
C INITIALIZE EPS
   EPS = .00001
C
C ENTER RECOVERY MODE
C
CALL ENTSR(IROLD,1)
C
10 CALL XMPL(X,EPS,10)
   IF (NERRO(NERR) .NE. 2) GO TO 20
C
C ERROR IN XMPL. INCREASE EPS.
C
   EPS = 10.*EPS
C
C TEST IF EPS TOO BIG
C
   IF (EPS .GT. EPSMAX) GO TO (failure)
C
C TURN OFF THE ERROR STATE AND TRY AGAIN
C
   CALL ERROF
   GO TO 10
C
C NO ERROR. PROCEED
C
20 ...

Error Recovery in Nested Subprograms

The error-handling mechanism has been constructed to deal with errors that occur when a
program calls on a lower-level subprogram and an error occurs in the called subprogram.
Because you may be unaware of the call to the lower-level subroutine, the PORT convention
assists you. It specifies that the outer program enter the recovery mode before it calls a
lower-level routine. Furthermore, when the outer program returns from that routine, it
tests to find whether or not an error has occurred. If so, the outer program reinterprets the
error for you.

The aforementioned subprogram ENTSR is used to enter the recovery mode (and store the
state of the recovery mode that had been in effect). The subprogram RETSR (return and
set recovery) is used when, after returning from the lower level subprogram, you wish to
restore the mode in effect at entry. Thus, the outer routine will contain the sequence:

   CALL ENTSR(IROLD,1)

This results in a call to the lower-level subprogram, followed by a test to see if any errors
have occurred and, if so, a reinterpretation of those errors.
CALL RETSR (IROLD)

RETSR acts as a “safety” exit gate. Since multiple errors are illegal, RETSR evaluates the situation and allows you to return to the calling program only if

1. There are no outstanding errors,

2. Or if the restored mode is the recovery mode that directs the calling program to be responsible for checking errors.

To avoid having multiple errors outstanding, it is a fatal error to call SETER or ENTSR while in the error state.

Other Error-handling Routines

EPRIN can be called to print the last error message if one exists.

FDUM is the dump routine called by SETER when a fatal error is encountered. Ideally the dump routine should provide a symbolic dump that lists for each active subprogram the names and values of all its variables, as well as the name of the calling routine.

Because it is impossible to write a portable FORTRAN subprogram to print symbolic dumps in an arbitrary FORTRAN environment, the package includes only a dummy version of the routine (just a RETURN statement). You should replace this version with a local version. If a full symbolic dump is unavailable, you should at least provide a traceback routine.
SPPS
an NCAR System Plot Package Simulator

The National Center for Atmospheric Research
Scientific Computing Division

NCAR is sponsored by the National Science Foundation
SPPS - AN NCAR SYSTEM PLOT PACKAGE SIMULATOR

CONTENTS

1. INTRODUCTION .............................................. 1
2. COORDINATE SYSTEMS ....................................... 2
3. COORDINATE-SYSTEM MAPPING FUNCTIONS ................. 4
4. SPPS SUBROUTINES .......................................... 5
   4.1 CLSGKS .................................................. 5
   4.2 CURVE (PX,PY,NP) ..................................... 5
   4.3 FLUSH .................................................. 5
   4.4 FL2INT (PX,PY,IX,IY) .................................. 5
   4.5 FRAME ................................................ 5
   4.6 FRSTPT (PX,PY) ......................................... 5
   4.7 GETSET (FL,FR,FB,FT,UL,UR,UB,UT,LL) ............... 5
   4.8 GETSI (IX,IY) .......................................... 6
   4.9 GETUSV (VN,IV) ........................................ 6
   4.10 LINE (X1,Y1,X2,Y2) ................................... 6
   4.11 MXMY (IX,IY) .......................................... 6
   4.12 OPNGIKS .............................................. 6
   4.13 PLOTIT (IX,IY,IP) ..................................... 6
   4.14 POINT (PX,PY) ......................................... 7
   4.15 POINTS (PX,PY,NP,IC,IL) ............................ 7
   4.16 PWRIT (PX,PY,CH,NC,IS,IO,IC) ...................... 7
   4.17 SET (FL,FR,FB,FT,UL,UR,UB,UT,LL) ................. 8
   4.18 SETI (IX,IY) .......................................... 8
   4.19 SETUSV (VN,IV) ....................................... 8
   4.20 VECTOR (PX,PY) ....................................... 10
   4.21 WTSTR (PX,PY,CH,IS,IO,IC) ........................ 10
5. CONVERSION CONSIDERATIONS .............................. 11
   5.1 INTEGER COORDINATES ................................. 11
   5.2 THE ROUTINE POINTS .................................. 11
   5.3 OPTION-SETTING ROUTINES .............................. 12
   5.4 INTENSITY AND COLOR .................................. 12
   5.5 SPOT SIZE ............................................. 13
   5.6 CONDITION-ACCESS ROUTINES ........................... 14
1. INTRODUCTION

The SCD Graphics Project decided some time ago to abandon the so-called 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; in the near future, graphics users will be 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 are used to "open" and "close" GKS. The former prepares GKS to receive SPPS output and must be called prior to calling any SPPS routine. The latter is called to close GKS after it has received output from SPPS.
- 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, PSSCALE, 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 one to map values from any NSPP/SPPS coordinate system to any other have been added.
- Two new routines, GETUSV and SETUSV, have been implemented in order to allow one to access "user state variables."

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 proper and are not described here.

The flash-package routines FLASH1, FLASH2, FLASH3, and FLASH4 are to be implemented and will be advertised when they are ready.
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:

  \[
  \begin{align*}
  \text{IMX} &= \text{IFIX}(32767 \times \text{FLOAT}(\text{IPX}-1)/(2 \times \text{MX} - 1)) \\
  \text{IMY} &= \text{IFIX}(32767 \times \text{FLOAT}(\text{IPY}-1)/(2 \times \text{MY} - 1))
  \end{align*}
  \]

  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:

  \[
  \text{IPX} = \text{IL} + \text{IFIX}((\text{RUX} - \text{UL})/(\text{UR} - \text{UL}) \times \text{FLOAT(IR} - \text{IL}))
  \]

  The logarithmic mapping of RUX into IPX is defined as follows:
IPX = IL + IFIX((ALOG10(RUX)-ALOG10(UL))/
+ (ALOG10(UR)-ALOG10(UL))*FLOAT(IR-IL))

The linear and logarithmic mappings of RUX 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:

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

Fractional coordinates may only be used as the first four arguments of a call to the routine
SET.
3. 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:

- CFUX(RX)
- CPFX(IX)
- KFMX(RX)
- KPMX(IX)
- CFUY(RY)
- CPFY(IY)
- KFMY(RY)
- KPMY(IY)
- CMFX(IX)
- CPUX(IX)
- KFPX(RX)
- KUMX(RX)
- CMFY(IY)
- CPUY(IY)
- KFPY(RY)
- KUMY(RY)
- CMUX(IX)
- CUFX(RX)
- KMPX(IX)
- KUPX(RX)
- CMUY(IY)
- CUFY(RY)
- KMPY(IY)
- KUPY(RY)

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.
4. SPPS SUBROUTINES

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).

4.1 CLSGKS. Closes GKS after output by SPPS.

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

4.2 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.

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

GKS considerations: None.

4.4 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

\[
\text{CALL FL2INT (PX,PY,IX,IY)}
\]

is the same as that of the two statements

\[
\begin{align*}
IX &= \text{KUMX(PX)} \\
IY &= \text{KUMY(PY)}
\end{align*}
\]

GKS considerations: None.

4.5 FRAME. Advances to a new frame.

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

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

\[
\text{CALL FRSTPT (PX,PY)}
\]

is equivalent to the statement

\[
\text{CALL PLOTIT (KUMX(PX),KUMY(PY),0)}
\]

GKS considerations: See PLOTIT.

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

4.8 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.

4.9 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.

4.10 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.

4.11 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.

4.12 OPNGKS. Opens GKS for output by SPPS.

GKS considerations: OPNGKS calls GOPKS to open GKS, GOPWK to open a metacode workstation, and GACWK to activate the workstation.

4.13 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.

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.

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

```
CALL POINT (PX,PY)
```

is equivalent to the statements

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

GKS considerations: None.

4.15 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. Other text attributes (character orientation, color, etc.) may be altered by the user prior to calling POINTS.

4.16 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.

4.17 SET (FL,FR,FB,FT,UL,UR,UB,UT,LL). Allows one 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 -- x linear, y linear
2 -- x linear, y logarithmic
3 -- x logarithmic, y linear
4 -- x logarithmic, y logarithmic

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

\[ FX = FL + (UX - UL) \left/ (UR - UL) \right. (FR - FL) \]

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

\[ FX = FL + (\log_{10}(UX) - \log_{10}(UL)) / (\log_{10}(UR) - \log_{10}(UL)) + (FR - FL) \]

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 one 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 silly 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.

4.18 SETI (IX,IY). Allows one 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 \(2^{**}IX-1\), inclusive, and plotter y coordinates are assumed to lie in the range from 1 to \(2^{**}IY-1\), inclusive.

GKS considerations: None.

4.19 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.

Following is a list of the user state variables:
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:

\[
\begin{align*}
RI &= \frac{IR}{\max(IR,IG,IB,1)} \times \frac{IN}{10000} \\
GI &= \frac{IG}{\max(IR,IG,IB,1)} \times \frac{IN}{10000} \\
BI &= \frac{IB}{\max(IR,IG,IB,1)} \times \frac{IN}{10000}
\end{align*}
\]

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, one should 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.

<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 '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: When the metacode unit number is changed by a call to GETSET, the current metacode workstation is closed and deactivated and the new one is opened and activated. (It is not clear at this time whether one will be able to resume output of metacode to a previously used metacode unit; check with a consultant for current information.) When the parameter 'IN' is set, the aspect source flags for the polyline, polymarker, text, and fill-area color indices are reset to "individual," the color indices themselves are all given a value II between 1 and 'IM', and the color representation for color index II is redefined on all open workstations to obtain the desired effect. When the parameter 'II' is set, the aspect source flags mentioned above are forced to "individual," and the color indices are given the value 'II'. When either of the parameters 'LW' or 'MS' is reset, the appropriate aspect source flag is set to "individual" and the appropriate GKS parameter is set.

4.20 VECTOR (PX,PY). Generates a pen-down move to the point (PX,PY), in the user coordinate system. Used in conjunction with FRSTPT (which see, 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.

4.21 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 counter-clockwise 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.
5. CONVERSION CONSIDERATIONS

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

5.1 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 statement

\texttt{CALL POINT (IX, IY)}

can be rewritten as the SPPS statement

\texttt{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

\texttt{CALL SET (IL, IR, IB, IT, ...)}

becomes the SPPS statement

\texttt{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

\texttt{CALL GETSET (IL, IR, IB, IT, ...)}

may be replaced by the SPPS statements

\texttt{CALL GETSET (VL, VR, VB, VT, ...)}

\texttt{IL = KFPX(VL)}

\texttt{IR = KFPX(VR)}

\texttt{IB = KFPY(VB)}

\texttt{IT = KFPY(VT)}

5.2 THE ROUTINE POINTS. The fourth argument of POINTS has been redefined. The NSPP call

\texttt{CALL POINTS (PX, PY, NP, 0, IL)}

may be left unchanged, but the NSPP call

\texttt{CALL POINTS (PX, PY, NP, +, IL)}

may be replaced by the SPPS statements

\texttt{CALL POINTS (CPFX(PX), CPFX(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 polyl markers 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.

5.3 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.

5.4 INTENSITY AND COLOR. The NSPP call

CALL OPTN ('IN',IN)

allowed one 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, one needs 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, 1:0:1 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, one 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, one sets the color to red and draws a line and then resets the color to blue and draws
another line, one can expect to get the obvious - a red line and a blue line. On devices having a screen that is continually 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 Dicom or for a pen plotter. The SPPS call

```
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 II to have the new value MOD(II,IM)+1 and uses that value for drawing subsequent objects. Note that 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 one sets ‘IM’ to 3 and draws four objects - the first and third in red, and the second and fourth in blue. If one 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, one 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

```
CALL SETUSV ('IR',1)
CALL SETUSV ('IG',0)
CALL SETUSV ('IB',0)
CALL SETUSV ('IN',5000)
CALL GETUSV ('II',I1)
```

define a half-intensity red and retrieve the color index associated with it in I1. Thereafter, one can use the SPPS call

```
CALL SETUSV ('II',I1)
```

to request that the same half-intensity red be used. If necessary, then, one can define, up front, all the colors/intensities one is 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.

5.5 SPOT SIZE. The NSPP call

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

set 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 Dicom. The SPPS call

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

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

5.6 CONDITION-ACCESS ROUTINES. The NSPP routines SETCND and GETCND do not have SPPS counterparts. The NSPP call

\texttt{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

\texttt{CALL SETCND ('YT',n)}

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

\texttt{CALL GETCND ('MU',n)}
\texttt{CALL SETCND ('MU',n)}

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

\texttt{CALL GETUSV ('MU',n)}
\texttt{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 a consultant to determine the current situation.
Elements and Format of the NCAR CGM Metafile

The National Center for Atmospheric Research
Scientific Computing Division

NCAR is sponsored by the National Science Foundation
The NCAR CGM Metafile

Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Record Formatting and Datatypes within NCAR</td>
<td>1</td>
</tr>
<tr>
<td>Class 1</td>
<td>2</td>
</tr>
<tr>
<td>Class 2</td>
<td>3</td>
</tr>
<tr>
<td>Class 3</td>
<td>3</td>
</tr>
<tr>
<td>Class 4</td>
<td>3</td>
</tr>
<tr>
<td>Class 5</td>
<td>3</td>
</tr>
</tbody>
</table>
The Elements and Format of the NCAR CGM Metafile

Introduction

This document describes the CGM-based NCAR metafile, a metafile that is being adopted for use with GKS systems within SCD and NCAR.

The elements of the NCAR metafile comprise a proper subset of the elements of the ANSI/ISO CGM (Computer Graphics Metafile) standard, as described in the document:

ISO DIS 8632
Information Processing Systems

Computer Graphics

Metafile for the Storage and Transfer of Picture Description Information

This document also describes the physical record format to be used with SCD-supported metafile software.

Record Formatting and Datatypes Within NCAR

The NCAR CGM metafile uses the Binary Encoding described in the CGM standard text. However, the CGM standard text section on Binary Encoding only describes the bit stream that comprises an extended metafile; it does not describe how the bit stream is bound into physical records.

Experimentation at NCAR indicates that fixed-length records have great advantages in a collection of diverse computer systems. Furthermore, it is advantageous to specify the record length as an integral number of words on as many computers as possible. A value of 11520 bits, or 1440 8-bit bytes, serves as a reasonable compromise between the requirements and capabilities of a range of systems from microcomputers to supercomputers. This value was previously used in the record structure of the old (non-CGM) NCAR metacode.

Because of the necessity of mixing metacode with other types of data within a file, a 4-byte control field is reserved at the beginning of each record. The first 16 bits is a data count that indicates the number of bytes in the remainder of the record that actually contain useful data. The next 4 bits is the datatype identifier field. Currently defined datatypes include header (0100), NCAR-formatted printer (1000), and old metacode (0010). One new type, extended metacode (0011), is defined to service the extended CGM metafile.

The zero-byte terminating record of the old metafile is of little use, and, in fact, has caused problems in the design logic of metafile editors. Because of the difficulties caused by the zero-byte terminating record, no SCD-supported metafile generators will generate zero-byte records. The metafile is a single binary file terminated with an EOF (End Of File) mark.

The single bit following the 4-bit datatype identifier will continue to be used as a "new-frame bit." When the new-frame bit is set, the data record in which it is contained is the first record of a new picture. This formatting feature does not prompt any graphical action; it is the BEGIN PICTURE and END PICTURE instructions that identify the boundaries of the
graphical contents of a picture. You will find the new-frame bit very useful for non-interpretive processing software such as metafile editors that extract whole frames from metafiles, split metafiles, and merge and concatenate metafiles.

The new-frame bit as defined above was also defined for the old (non-GCM) NCAR metacode. The two bits following the new-frame bit are now defined for the new CGM metacode only. The first of the newly defined bits is the "begin metafile" bit, which marks the first record of a metafile and hence the record containing the picture descriptor. The bit that follows is the "end metafile" bit, which declares that the record is the last record of a metafile and that it contains an END METAFILE instruction. These two bits allow multiple metafiles to reside on a single physical medium, and to be easily recognized by non-interpretative processes such as editors.

The control bits are summarized in their order as follows:

- 1-16: the byte count for the record.
- 17-20: the datatype flag.
- 21: the new-frame bit.
- 22: the begin metafile bit.
- 23: the end metafile bit.

Although instructions may cross record boundaries, an individual operand component such as a coordinate or a color index may not be split across record boundaries. The 16-bit alignment requirement makes it impossible to split operand components across record boundaries, since the default precision value of the metafile is 16 bits. This default value applies to coordinate precision, color index precision, and so on. Splitting operand components across record boundaries would be possible if precision were increased beyond 16 bits.

The CGM defines a metafile descriptor that contains no pictorial information, but includes descriptive information that aids in interpreting the metafile. The NCAR metafile descriptor will consist of one or more fixed-length data records that look like any other metafile data records. There is only one metafile descriptor, and it occurs at the beginning of the metafile. The new-frame bit of the 32-bit control field is set to zero in metafile descriptor records. The metafile descriptor in the NCAR CGM is that sequence of contiguous records at the beginning of the metafile up to (but not including) the first record with the new-frame bit set. As previously mentioned, the NCAR CGM is a proper subset of those elements mentioned in the CGM standard. The following list contains the elements in the CGM standard that are not presently supported in the NCAR CGM standard:

**Class 1**

- VDC TYPE
- INTEGER PRECISION
- REAL PRECISION
- INDEX PRECISION
- COLOUR PRECISION
- COLOUR INDEX PRECISION
- MAXIMUM COLOR INDEX
- COLOUR VALUE EXTENT
The NCAR Graphics Package

FONT LIST
CHARACTER SET LIST
CHARACTER CODING ANNOUNCER

Class 2

SCALING MODE
LINE WIDTH SPECIFICATION MODE
MARKER SIZE SPECIFICATION MODE
EDGE WIDTH SPECIFICATION MODE

Class 3

VDC REAL PRECISION
AUXILIARY COLOUR
TRANSPARENCY

Class 4

DISJOINT POLYLINE
RESTRICTED TEXT
APPEND TEXT
POLYGON SET
RECTANGLE
CIRCLE
CIRCULAR ARC 3 POINT
CIRCULAR ARC 3 POINT CLOSE
CIRCULAR ARC CENTRE
CIRCULAR ARC CENTRE CLOSE
ELLIPSE
ELLIPTICAL ARC
ELLIPTICAL ARC CLOSE

Class 5

CHARACTER SET INDEX
ALTERNATE CHARACTER SET INDEX
EDGE BUNDLE INDEX
EDGE TYPE
EDGE WIDTH
EDGE COLOUR
EDGE VISIBILITY
The NCAR
Computer Graphics Metafile Translator

by: John X. Humbrecht

June 17, 1986

The National Center for Atmospheric Research
Scientific Computing Division
NCAR is sponsored by the National Science Foundation
SECTION I:
INTRODUCTION
This document describes the operation of the NCAR Computer Graphics Metacode (CGM) Translator. The CGM used by the NCAR Translator is defined by Part 3 of the document:

ISO/DIS 8632
Information Processing Systems
Computer Graphics
Metafile for the Storage and Transfer
of Picture Description Information
Version 7.4

The NCAR CGM translator consists of three programs and a number of support files. Supplied by NCAR are the portable Fortran source code for the programs and character support files. Installation requires a few steps:

1. Write installation dependent routines.
2. Determine how to declare the font file and device description file names to the Translator. This has been implemented as command lines, fixed names in common blocks and environment variables in 4.2 BSD UNIX.
3. Determine how to provide the Metafile name to the translator. The same options hold here as in the previous point.
4. Create executables for all necessary programs.
5. Create the necessary character versions of the grapcap files (device descriptions) for your devices.
6. Create the translator readable font and device descriptions from the character files.
7. Run the translator.

The installation dependent routines are described in a later section of this manual. They deal with I/O and bit manipulation. These routines should require only a few days to construct. Some of them are the same as those required by the GKS 0a environment.

The implementer has a few options for deciding the method of informing the translator as to the font and device:
- The command line may be used to communicate the names of the support files. In this case the routine FNDARG will parse for the names and pass them to the REDDEV and REDFNT routines.
- They can be hard wired into the variables GRCAPN and FNCAPN defined in Block Data TRNDAT. This is the least desirable option.
- On 4.2 BSD UNIX I use environment variables to define the file names. The variables GRCAPN and FNCAPN are set to the environment variable names and the local open routines handle the use of environments.
- Have the routines REDDEV and REDFNT query the user for the appropriate names.

The present system only contains one font file.

Providing the Metafile name to the translator can be handled in a way similar to the support file options described above. The routine FNDARG controls the opening of the Metafile. The installations that I have participated in all use the command line to communicate the Metafile name to the translator. The routine FNDARG is then instrumented to parse the command line and open the file.

The contents of this manual are primarily oriented toward the installer of the software and the writers of GRAPHCAPs and FONTCAPs.
The creation of a GRAPHCAP requires extensive knowledge of computer graphics display devices. Familiarity with the operating manual and functions of the device are necessary to provide the information required in a GRAPHCAP to accurately drive the device. Users of the translator may find the GRAPHCAP section useful for tailoring an existing GRAPHCAP to their own needs.

Many times a new device may defined by modifying an existing GRAPHCAP for a device with similar characteristics. This quite often saves time and effort over creating the description from scratch.

This document is divided into five sections:

I. An overview of the translator.
II. The Graphcap Forms manual describing the means to modify and add to the device driver library.
III. The Fontcap Forms manual describing the means to modify and add to the software font generator library.
IV. The installation requirements. This section covers implementation topics including system dependent routines and command interfacing.
V. A description of the error codes produced by the translator during execution.

The operation of the translator depends on translator readable Graphcaps and Fontcaps, a functioning translator, CGM or NCAR System Plot Package metacode, and the correct display devices.

To generate translator readable cap files, you must pass a source version through the appropriate preprocessor. There is one for Fontcaps and one for Graphcaps. The installation of the preprocessors requires routines from the local implemented set described in Section IV. The conversion process for cap files (Graphcap and Fontcap) is done at installation and thereafter, when you modify or add to the device or font libraries (in this case only the new or changed files need to be converted).

To select color using NCAR System Plot Package and this generation of translators use a character string naming a decimal position in the device color table (or intensity table if the device is monochrome). This is accomplished by sending a 3 digit ASCII string for the color in OPTN routine color selection. Color 000 is the background and will not be visible unless drawing over an object of another color. The example below will change from the current color to color 3 on the destination device:

\[
\text{CALL OPTN ('CO', '003')}
\]

Setting the color on a device with no such capability will have no effect. Note: the color OPTN discussion pertains to the use of the NCAR System Plot Package. If you are an NCAR GKS user there are other means of selecting colors.

If you are running the translator on an SCD supported computer, Appendix 2 will provide information about its operation.

The GKS features of Polygon clipping and Pattern filling are not supported in this version of the NCAR CGM translator. Polygon style of other than hollow is only supported on devices which provide hardware fill capabilities. Later versions of the translator will include these features. All other features which may be generated by the NCAR GKS 0a are supported.
SECTION II:
GRAPHCAP FORMS
1. INTRODUCTION

This section describes the use of the GRAPHCAP documents. A GRAPHCAP document is a table describing the characteristics of a graphics device.

This section is oriented toward the user who is familiar with the operation and instruction set of the target computer graphics display device. Casual users will find that generating a graphcap from scratch is not an easy task. However, modifying the graphcap to produce a different color table, change the frame finished prompt, modify the picture orientation or select a different line type are tasks easily performed by the user. This allows users to customize the translator to their own requirements without involving programming personal.

In its original form the document consists of a set of keywords followed by the definition of that operation on a selected graphics device. A keyword does not have to be in every document. Usually, if the device does not support the option, the keyword is left out of that document. If a keyword is present but not defined, it is equivalent to the keyword not being present. In both of these cases the default definition (if any) will be used. A keyword must be contained on one line.

To create or modify a GRAPHCAP use your favorite text editor, provided that the editor does not insert any of its own control characters. After creating or modifying a GRAPHCAP document, it must be processed by the GRAPHCAP processor before it can be used as input to the translator.

The keyword definitions consist of four types:

1. Logical, valued TRUE or FALSE. The value is stored, for the target machine, in a FORTRAN type LOGICAL.
2. Decimal valued. A positive or negative decimal value, separated by blanks. The value is stored, for the target machine, in a FORTRAN type INTEGER.
3. Floating point valued. A floating point value must contain a decimal point. Floating point values are separated by blanks. The value is stored, for the target machine, in a FORTRAN type REAL.
4. String type. A string consists ASCII characters plus special control values which control the placement of addresses, coordinates, and counts within instruction strings. ASCII characters taken from the ASCII table in Appendix 1. Each String value is separated by one or more blanks or a new line. The value is stored, for the target machine, in the lower 8 bits of a FORTRAN type INTEGER.

The other parameters allowed in the String set are enumerated below and in Appendix 1.

- INT an Integer value. At times it is necessary to represent integer values in these strings. Each 8 bits may be loaded with an integer value from 0 to 255 by using the prefix INT before each integer. For example:
  
  \[37 = \text{INT37} \]
  \[200 = \text{INT200} \]

- MAD - Used when defining a COLOR/INTENSITY map table instruction start. It declares that a Map Address should be inserted at this position in the instruction. See the DEVICE_MAP_INSTRUCTION_START discussion for examples.

- VC - Used to indicate that a vector count is required at this position in the instruction.

- XYC - Indicates that an X, Y coordinate pair is required at this point in the instruction.

All keyword definitions may range from null to a maximum count specified by that keyword. Keyword definitions are allowed a maximum of 80 columns per line, however they may cross line boundaries. A new line is equivalent to a space for separating definition entities.

To understand and change the BUNDLE class, please refer to the Graphical Kernal System workstation states for the appropriate entry. These are found in the GKS 7.2 standards document and are not further discussed here. The BUNDLE class must be included and defined in every GRAPHCAP used. The blank Graphcap form defines the Bundle entries as required by GKS. A user may make nonstandard definitions by changing these entries. Please comment any changes made to Bundles, this will alert others to non-standard features in the Graphcap.
When defining a new device it may be easiest to start with an existing GRAPHCAP, which is similar, and work from there. For example, many of the Tektronix emulators only require changes to the DEVICE_GRAPHIC_INIT and DEVICE_TEXT_INIT keyword definitions to operate properly.

A blank form is included with the GRAPHCAP files, please use this as a starting point for generating a new device form.

Keywords may be in any order, the groupings below are for description only.

When reading this section it is best to get a copy of a Graphcap for a device you are familiar with and use it as an example of the keyword definitions.
2. FORMATTING and ENCODING

The process of creating coordinates, color/intensities, and other parameters which are part of a device instruction is normally not simple. Many devices require bits from various parts of the data values be packed together to form the device instruction parameters. For example the Tektronix 40XX series requires that the first word of a device coordinate parameter contain 5 bits from the upper part of the Y coordinate packed with some flagging information. Other devices require parts of the X and Y coordinate be packed together. Describing this formatting information to the translator requires that a table be filled in which contains the bit positions for extracting from input data and inserting into device parameters.

Once the data is successfully packed, many devices require a second massaging prior to sending to the device. Encodings take the bit stream created by the Formatting process and convert to another bit stream which is sent to the device. The Tektronix 41XX terminal series requires that the bit stream from the formatting process be converted into a set of printable characters.

Many devices require that parameters be converted through these two steps. This discussion will explain the conversion process used for all keywords which contain FORMAT and ENCODING.

2.1 The Formatting Process

The users should think of the formatting process as consisting of input data composed of an array of integers which are mapped into an output stream of bits.

The input words are addressed from the right, traveling left. Each word in the array is addressed as in a FORTRAN array, that is the first element of the array is at position 1. The first bit of each element in the array is at position 0 (this is the rightmost bit, values increase to the left).

The output stream is a string of bits, there is no regard for word boundaries. The stream is addressed from left to right. The first bit is at Address 0.

Data is extracted from input words and inserted into the output stream according to the information given in the formatting table. This table contains 4 entries per row. Each row defines a separate operation of extracting from an input word and inserting into the output stream. Each row contains a start_bit, bit_count, data_type and data_value. All entries must be defined.

A. Start_bit-The index to start at in the output string.
B. Bit_count-the number of bits to be transferred.
C. Data_type-determines where input data originates for the extraction step. Valid values of the data_type are:

   -1 This data_type will extract bits from the data_value and inserts into the output stream. It then sends the stream to the encoder, this has the effect of clearing the stream (the output stream is disposed to the encoder and then to the device). You should start addressing the output stream from the left edge, address 0 (zero), with the next row of the format table (if you have another entry). -1 is typically used with the ASCII encodings to signal the end of a number to be encoded and to insert a separator in the device stream. The bit_count must be divisible by 8 (fill 8 bit bytes).

   0 Flag value. Move the contents of the data_value field to the output stream.

   > Use the addressed input word.

D. Data_value, its use is determined by the type of operation requested by data_type.

- A data_type of -1 uses the data_value as the input word. Bit_count bits are moved from the data_value to the output stream.
- A data_type of 0 (zero) uses the data_value as the input word. Bit_count bits are moved from the data_value to the output stream.
- For data_type's greater than 0, move bits from the addressed bit position in the input word to the output stream. A 10 in this position would extract bits starting at bit position 10 (0 at right edge of the word) for bit_count bits towards the right.
In most cases you will not know how many input words will be received. So how can you enter
enough data values greater than zero to cover the input stream? First if you request too many data
values unexpected and unpredictable things will happen. However, you do not need to specify enough
entries for all input data. The format table will cycle until all input data has been processed. Suppose
you had a table defining XY coordinates. The X's and Y's are packed together in some strange way (Tek
4010 for example). You just define how the first two words are packed (word 1 is the X and word 2 is
the Y). The table will cycle until all the input words have been processed. For a color table you may
have 3 entries for each color definition, so define how those 3 entries are constructed and the table will
cycle for all the rest of the data.

I will give an example for the TEK 4010 coordinate formatting to demonstrate how to construct a
format table. All keywords related to formatting will contain examples specific to their subject. Each
line describes an 8 bit byte which is to be sent to the device. The letters represent; P for parity and X
and Y for the X and Y coordinates. The digit following X or Y is the bit being referred to in the coordi
nate as described for the input data indexing above. The 0's (zero) and 1's are required by those bit
positions. This description is taken from a Tektronix terminal Programmers manual.

P 0 1 Y9 Y8 Y7 Y6 Y5
P 1 1 Y4 Y3 Y2 Y1 Y0
P 0 1 X9 X8 X7 X6 X5
P 1 0 X4 X3 X2 X1 X0

The table to describe this will appear as follows;

<table>
<thead>
<tr>
<th>bit_start</th>
<th>bit_count</th>
<th>data_type</th>
<th>data_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

More examples are provided in each FORMAT type keyword.

2.2 The Encoding Process
The Encoding process takes the output stream from the formatter and performs another massaging off
the data prior to sending to the device. The encoding is defined by one of the decimal values described
below.

0 - binary encoding; No change to the formatted data.
1 - ASCII decimal encoding; The bit stream from the formatter is translated into a ASCII string
representing the decimal equivalent of the string. The bits 100 would be converted to an ASCII
8 (decimal 56).
2 - ASCII hexadecimal encoding; The bit stream from the formatter is translated into a ASCII
string representing the hexadecimal equivalent of the string. The bits 1111 would be converted
to an ASCII F (decimal 70).
3 - ASCII octal encoding; The bit string from the formatter is translated into a ASCII string
representing the octal equivalent of the string. The bits 1000 would be converted to two ASCII
characters 10 (decimal 49 and decimal 48).
4 - ASCII Tektronix encoding; This type of encoding is used by the Tektronix 41xx class of dev
ices.
5 - ASCII Real; This encoding translates a bit stream into a printat; floating point number.
The conversion from the bit stream, which is interpreted as an whole number, depends on the
corresponding FLOATING_INFO keyword. The FLOATING_INFO type keywords provide 4 real numbers to the encoder. The first two are the minimum and maximum values expected from the bit stream. The last two specify the minimum and maximum floating point values to generate. A linear mapping is performed from input to output.
3. WORKSTATION INITIALIZATIONS

The following keywords describe the strings used to set the device into graphics for text mode. Graphics mode is the state where graphics instructions are received and understood. Text mode is used for normal communications and editing.

**DEVICE_GRAPHIC_INIT**

Description: Enter graphics mode. Sent at invocation of the translator.

Type: String.

Default: None

Maximum number of entries: 100

**DEVICE_TEXT_INIT**

Description: Enter text mode. Sent at termination of the translator.

Type: String.

Default: None

Maximum number of entries: 100

**DEVICE_BATCH**

Description: Logical value indicating whether the translator sends the USER_PROMPT and waits for keyboard Carriage return. This operation happens at completion of a picture. Batch devices may include laser printers and film devices. Non-batch devices may be graphics display terminals. Batch devices do not prompt for user intervention, they automatically proceed to the next frame.

Type: Logical

Default: FALSE, user intervention is required to advance pictures.

Maximum number of entries: 1

**DEVICE_ERASE**

Description: String sent to the device when the display is to be cleared.

Type: String

Default: None

Maximum number of entries: 20

**DEVICE_CURSOR_HOME**

Description: String sent to the device when the cursor should be positioned at the home position. The translator considers home as the upper left hand corner of the picture drawing surface.

Type: String

Default: None

Maximum number of entries: 50

**USER_PROMPT**

Description: The string sent to the device indicating to the user that the picture on the display surface is finished. At this point some device dependent action (usually a carriage return) is required to continue. This string is used only if DEVICE_BATCH is set FALSE. Typically the prompt is displayed in Graphics mode. On small single page plotters you may wish to terminate graphics mode in this string and then send the prompt. This has the effect of sending the prompt to the...
CRT connected to the plotter but not displaying it on the plotter.

Type: String.
Default: None

Maximum number of entries: 50
4. WORKSTATION DRAWING SPACE

This set of keywords describes the drawing surface available on the workstation. The picture can be drawn in any orientation or transposition by properly changing the lower left and upper right coordinates and perhaps interchanging the order of coordinate processing as defined by the DEVICECOORDFORMAT.

If your device has lower left X, Y at 0, 0 and upper right at 1000, 1000, you could draw the picture upside down by setting the lower left coordinates to 0, 1000 and the upper right coordinates to 1000, 0.

DEVICECOORDLOWERLEFTX

Description: The device X coordinate of the lower left corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

DEVICECOORDLOWERLEFTY

Description: The device Y coordinate of the lower left corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

DEVICECOORDUPPERRIGHTX

Description: The device X coordinate of the upper right corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

DEVICECOORDUPPERRIGHTY

Description: The device Y coordinate of the upper right corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

DEVICECOORDXOFFSET

Description: A device coordinate value added to X addresses before they are sent to the device. The translator attempts to center the plot based on the lower left, upper right values. When the coordinate space is asymmetric (x coord interval is not the same as the y coordinate interval) offsets will help properly place the frame. Another use is to offset the frame to a specific side of the display.
Type: Decimal
Default: 0 (zero)
Maximum number of entries: 1

DEVICECOORDYOFFSET

Description: A device coordinate value added to Y addresses before they are sent to the device. See DEVICECOORDXOFFSET discussion.
Type: Decimal
Default: 0 (zero)
Maximum number of entries: 1

DEVICE_COORD_XSCALE
Description: In some cases a X unit does not equal a Y unit. This keyword allows a scale factor to be introduced into the device coordinate computation stream. It will scale the X coordinates up or down by the specified amount.
Type: Floating point
Default: 1.0
Maximum number of entries: 1

DEVICE_COORD_YSCALE
Description: See the DEVICE_COORD_XSCALE discussion.
Type: Floating point
Default: 1.0
Maximum number of entries: 1

DEVICE_COORD_FORMAT
Description: The format used to organize the device coordinate data. The input array to this format is a set of coordinate pairs. See the discussion of formatting at the beginning of this section. Two examples follow. They describe the required bit positions as P for parity, Xn for a X bit and Yn for a Y bit. Parity is a don't care condition so those bits are skipped over in these examples. Lets try the AED512 7 bit binary encoding for the first example.

P X9 X8 X7 Y10 Y9 Y8 Y7 
P X6 X5 X4 X3 X2 X1 X0 
P Y6 Y5 Y4 Y3 Y2 Y1 Y0 

The format table will now appear as below:

<table>
<thead>
<tr>
<th>DEVICE_COORD_FORMAT</th>
<th>bit_start</th>
<th>bit_count</th>
<th>data_type</th>
<th>data_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>-1</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>-1</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

For the next example we will use the AED512 ASCII decimal encoding. This encoding uses a 16 bit Address and each coordinate is terminated by a blank. The X coordinate is first followed by the Y coordinate.

<table>
<thead>
<tr>
<th>DEVICE_COORD_FORMAT</th>
<th>bit_start</th>
<th>bit_count</th>
<th>data_type</th>
<th>data_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>-1</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>-1</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>
Type: Decimal rows with 4 elements per row.
Default: None
Maximum number of entries: 80 (20 rows * 4 columns)

DEVICE_COORD_ENCODING
Description: The encoding scheme used for device coordinates. See the Encoding discussion at the beginning of this section.
Type: Decimal in the range of 0 to 5.
Default: 0 (zero)
Maximum number of entries: 1

DEVICE_COORD_FLOATING_INFO
Description: The mapping from fixed point (integer) to output floating point. This option is used when the DEVICE_COORD_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the device.
Type: Floating point
Default: None
5. DEVICE VECTOR COUNTS

The vector count is a parameter which indicates the number of points (X, Y coordinate pairs) which are part of the current instruction. These keyword definitions are activated by the VC parameter which may be included in the LINE, MARKER and POLYGON instruction start classes.

DEVICE VECTOR COUNT_ENCODING

Description: The encoding scheme used for vector counts. See the Formatting and Encoding discussion at the start of this section.

Type: Decimal in the range of 0 to 5.
Default: 0 (zero)
Maximum number of entries: 1

DEVICE VECTOR COUNT_FLOATING_INFO

Description: The mapping from fixed point (integer) to output format floating point. This option is used when the DEVICE VECTOR COUNT_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the device.

Type: Floating point
Default: None

DEVICE VECTOR COUNT_FORMAT

Description: The format used to organize the vector count data. This formatter will receive only one input data word. See the discussion on Formatting and Encoding at the start of this section.

Example

This example encodes a 7 bit vector count. The index is right justified in the vector count parameter passed into the formatter. The P represents parity bit and V followed by a digit is a bit out of the vector count. Parity is a don't care condition and is skipped over. We are using the binary encoding scheme.

P V6 V5 V4 V3 V2 V1 V0

This format would be set up as follows:

<table>
<thead>
<tr>
<th>DEVICE VECTOR COUNT_FORMAT</th>
<th>bit_start</th>
<th>bit_count</th>
<th>data_type</th>
<th>data_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Type: Decimal rows with 4 elements per row.
Default: None
Maximum number of entries: 40 (10 rows * 4 columns)
6. DEVICE COLOR CAPABILITIES

This section describes the general color availability on a given device. For specific color operations refer to the LINE, MARKER and POLYGON primitives referenced in later parts of this section.

DEVICE_COLOR_AVAILABLE

Description: Flag to indicate the availability of color on a device. If the device color is TRUE then DEVICE_INDEX_RANGE_DEFINED keyword must be defined in the GRAPHCAP.

Type: Logical
Default: FALSE, no color on the device.

Maximum number of entries:

DEVICE_COLOR_INDEX_ENCODING

Description: The encoding scheme used for color indices. See the discussion on Formatting and Encoding at the start of this section.

Type: Decimal in the range of 0 to 5.
Default: 0 (zero)

Maximum number of entries: 1

DEVICE_COLOR_INDEX_FORMAT

Description: The format used to organize the color index data. The formatter will receive only one input data word. See the discussion on Formatting and Encoding at the start of this section.

Example

This example uses a four bit color index (0 to 15 decimal). The P represents parity bit 0's (zero) and 1's represent themselves and c followed by a digit is a bit out of the color index. Parity is a don't care condition and is skipped over. We are using the binary encoding scheme.

P 0 1 1 c3 c2 c1 c0

This format would be set up as follows:

```
<table>
<thead>
<tr>
<th>bit_start</th>
<th>bit_count</th>
<th>data_type</th>
<th>data_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
```

Type: Decimal rows with 4 elements per row.
Default: None

Maximum number of entries: 40 (10 rows * 4 columns)

DEVICE_COLOR_INDEX_FLOATING_INFO

Description: The mapping from fixed point (integer) to output format floating point. This option is used when the DEVICE_COLOR_INDEX_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the device.

Type: Floating point
Default: None
DEVICE_MAPAVAILABLE
Description: A flag indicating the availability of a user settable color/intensity map.
Type: Logical
Default: FALSE, no user settable map available.
Maximum number of entries: 1

DEVICE_MAP_INDEX_RANGE_MAX
Description: The size of the user settable intensity map. If the map has 256 entries (0 to 255) then use 256.
Type: Decimal
Default: 0 (zero)
Maximum number of entries: 1

DEVICE_MAP_INDEX_RANGE_DEFINED
Description: The range of colors defined by the default settings. If the device had entries 0 to 7 defined then use 8. This keyword must be defined if the device has color capability even though the color map is not settable.
Type: Decimal
Default: 0 (zero)
Maximum number of entries: 1

DEVICE_INTENSITY_ENCODING
Description: The encoding scheme used for intensity settings. See the Formatting and Encoding discussion at the start of this section.
Type: Decimal in the range of 0 to 5.
Default: 0 (zero)
Maximum number of entries: 1

DEVICE_MAP_INTENSITY_FLOATING_INFO
Description: The mapping from fixed point (integer) to output format floating point. This option is used when the DEVICE_MAP_INTENSITY_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the device.
Type: Floating point
Default: None

DEVICE_MAP_INTENSITY_FORMAT
Description: The format used to organize the device map intensity data. This formatter expects sets of 3 input data words
if in a multi-chrome mode and single input data words if in a mono-chrome mode. Observe the setting of the DEVICE_MAP_MODEL keyword for the mode used by this formatter. See the discussion on Formatting and Encoding at the start of this section.

Example

Assume a RGB model with 100 map entries. The formatter will have 300 words passed to it (3
per entry). The device uses 8 bits per intensity. We are using a binary encoding for this example.

<table>
<thead>
<tr>
<th>DEVICE_MAP_INTENSITY_FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit_start</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

All the intensity settings are treated the same so we only need one entry in the table to format all 300 settings.

The next example will also have 100 RGB map entries. The device uses an ASCII decimal encoding. Red and Green are separated by a blank. Green and Blue are separated by a period. Green and the Red of the next entry are separated by a colon.

<table>
<thead>
<tr>
<th>DEVICE_MAP_INTENSITY_FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit_start</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

Type: Decimal 4 entries per row.
Default: None
Maximum number of entries: 200 (50 rows * 4 entries per row).

DEVICE_MAP_INDIVIDUAL

Description: A flag that indicates a DEVICE_MAP_INSTRUCTION_START and TERMINATOR may need to be sent for each entry. If the flag indicates no need the DEVICE_MAP_INSTRUCTION_START and TERMINATOR are sent only once for map intensity streaming.

Type: Logical
Default: FALSE, DEVICE_MAP_INSTRUCTION_START and TERMINATOR are sent only once for map intensity streaming.
Maximum number of entries: 1

DEVICE_MAP_INSTRUCTION_START

Description: The string sent to the device to enable it to receive a new intensity map definition. The MAD String parameter is used in this string if Map Addresses are required by the device instruction.

Type: String
Default: None
Maximum number of entries: 30

DEVICE_MAP_INSTRUCTION_TERMINATOR

Description: Terminate the Color Map instruction.

Type: String
Default: None
Maximum number of entries: 10

**DEVICE_MAP_INIT**

Description: The intensity map used for initialization of the device prior to each picture being displayed. This data is relative to the device model selected via the DEVICE_MAP_MODEL and is given in device dependent intensity values.

When the Encoding is ASCII real (value 5) then the values given here must be integers in the range of the first two values of the DEVICE_MAP_FLOATING_INFO. The map intensities are mapped to the floating values when sent to the encoding process. This is necessary because this keyword allows only decimal input.

Type: Decimal

Default: None

Maximum number of entries: 768 \( (256 \times 3) \)

**DEVICE_MAP_MODEL**

Description: This keyword defines the type of intensity model used by the device. I have selected four types to include in the translator.

- 0 (zero) - Monochrome This model is used on monochrome devices (one color) which have variable intensities. A good example is a black and white film device.
- 1 - RGB Red, Blue, Green, a popular model on color devices. Intensities are given as 3-tuples.
- 2 - BGR Blue, Green, Red. NCAR has the unfortunate luck to support this model because of RAMTEK color devices.
- 3 - HLS The Hue, Lightness and Saturation model is much easier to understand than RGB and is used by TEKTRONIX color devices.

Type: Decimal in the range of 0 (zero) to 3.

Default: 1 (RGB)

Maximum number of entries: 1
7. LINE CONTROL

Used for drawing vectors and text. This set of instructions may also be used by the translator to position the device graphics cursor.

**LINE_DRAW_POLY_FLAG**

Description: Some devices support polylines, which are distinguished by the polyline instruction start followed by the coordinates for the entire line. Other devices do not support polyline which requires that each coordinate pair be accompanied by an instruction whether it be a move or draw.

This flag indicates the availability or lack of a polyline instruction on the device.

When a polyline instruction is available the LINE_DRAW_INSTRUCTION_START is used to initiate the polyline and the LINE_DRAW_INSTRUCTION_TERMINATOR ends the polyline.

When there is no polyline on the device the MOVE set is used for moving and the DRAW set is used for drawing.

Type: Logical

Default: FALSE, no polyline available on the device.

Maximum number of entries: 1

**LINE_DRAW_INSTRUCTION_START**

Description: The instruction which produces a line on the display surface. If the device has polylines then this instruction is sent at the start of a polyline.

Type: String

Default: None

Maximum number of entries: 20

**LINE_DRAW_INSTRUCTION_TERMINATOR**

Description: Terminate the draw instruction. This is also used to terminate the polyline instruction.

Type: String

Default: None

Maximum number of entries: 5

**LINE_MOVE_INSTRUCTION_START**

Description: Move the device graphics cursor. This string is not used when the device has polylines.

Type: String

Default: None

Maximum number of entries: 20

**LINE_MOVE_INSTRUCTION_TERMINATOR**

Description: Terminate the move instruction. This string is not used if the device has polylines.

Type: String

Default: None

Maximum number of entries: 5

**DASH_BIT_LENGTH**
Description: The number of VDC units used for each bit of a software dash line pattern. This dash pattern is defined in the Metacode or by the bundle tables. Modifying this allows all devices to generate a similar sized dash. The default dash patterns are found in the translator source file TRNDAT which is a BLOCKDATA routine. If you select GKS line indices 1-4 you will get one of these dash patterns.

Type: Decimal
Default: 100 there are 100 VDC units per dash pattern bit.
Maximum number of entries: 1

**LINE_COLOR_INSTRUCTION_START**
Description: The instruction to change the line color. A color index is sent after this string.
Type: String
Default: None
Maximum number of entries: 20

**LINE_COLOR_INSTRUCTION_TERMINATOR**
Description: Terminate the line color instruction.
Type: String
Default: None
Maximum number of entries: 5

**LINE_WIDTH_INSTRUCTION_START**
Description: Change the line width on the display surface.
Type: String
Default: None
Maximum number of entries: 10

**LINE_WIDTH_INSTRUCTION_TERMINATOR**
Description: Terminate the line width instruction.
Type: String
Default: None
Maximum number of entries: 10

**LINE_WIDTH_ENCODING**
Description: The encoding scheme used for line width values. See the discussion on Formatting and Encoding at the start of this section.
Type: Decimal in the range of 0 to 5.
Default: 0 (zero)
Maximum number of entries: 1

**LINE_WIDTH_FLOATING_INFO**
Description: The mapping from fixed point (integer) to output format floating point. This option is used when the LINE_WIDTH_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the
device.
Type: Floating point
Default: None

LINE_WIDTH_FORMAT
Description: The format used to organize the line width data. The formatter will receive one input data word.

Example
A sixteen bit binary line width is being sent to the device.

```
/* LINE_WIDTH_FORMAT */
bit_start bit_count data_type data_value
0 16 1 15
```

Type: Decimal 4 entries per row.
Default: None
Maximum number of entries: 20 (5 rows * 4 entries per row).

LINE_WIDTH_RANGE
Description: Two values giving the smallest and largest widths available in device units. Required if the LINE_WIDTH_INSTRUCTION_START is defined.
Type: Decimal
Default: None
Maximum number of entries: 2 (always required if this keyword is defined)

LINE_WIDTH_SCALE
Description: A value multiplied into the computed device line width units before it is sent to the device.
Type: Floating Point
Default: 1.0
Maximum number of entries: 1
8. MARKER CONTROL
This section controls the generation of polymarkers. The translator does not take advantage of hardware markers. Hardware markers were not accurate enough for our test users. The translator only uses software markers. The only instructions in this class are for changing color. The line instructions are used for drawing the markers. The color change instructions should be the same as for line color.

MARKER_COLOR_INSTRUCTION_START
Description: Change the marker color.
Type: String
Default: None
Maximum number of entries: 10

MARKER_COLOR_INSTRUCTION_TERMINATOR
Description: Terminate the color instruction.
Type: String
Default: None
Maximum number of entries: 10
9. POLYGON CONTROL

This section controls the generation of Polygons, closed objects which may be filled or hollow. The coordinates are encoded and formatted using the DEVICECOORD definitions. If there is no polygon instruction on the device then the polygon color instructions should be the same as the line color instructions.

POLYGON_INSTRUCTION_START

Description: If the device has a polygon instruction then this field is defined. If the field is not defined then line instructions are used to generate the outline of the polygon and fill style is hollow.

Type: String
Default: None
Maximum number of entries: 20

POLYGON_INSTRUCTION_TERMINATOR

Description: Terminator for the hardware polygon instruction.

Type: ASCII characters
Default: None
Maximum number of entries: 10

POLYGON_COLOR_INSTRUCTION_START

Description: The instruction to change polygon color.

Type: String
Default: None
Maximum number of entries: 10

POLYGON_COLOR_INSTRUCTION_TERMINATOR

Description: Terminate the polygon color instruction

Type: String
Default: None
Maximum number of entries: 10
10. WORKSTATION RASTER DRAWING SPACE

This set of keywords describes the raster drawing surface available on the workstation. The raster class of instructions is utilized when the CGM CELL ARRAY instruction is encountered. The picture can be drawn in any orientation or transposition by properly changing the lower left and upper right coordinates and perhaps interchanging the order of coordinate processing as defined by the RASTER_COORD_FORMAT. Of course the strings defined in RASTER_HORIZONTAL_INSTRUCTION class must function properly in the defined orientation.

If your device has lower left X, Y at 0, 0 and upper right at 1000, 1000, you could draw the picture upside down by setting the lower left coordinates to 0, 1000 and the upper right coordinates to 1000, 0.

RASTER_SIMULATE

Description: A device which does not support scan line instructions can be made to display a CELL ARRAY. When this option is set TRUE, the translator will attempt to send filled polygons to the device. Each polygon will represent one element of the CELL ARRAY. If your device does not support filled polygons then you will get quadrilaterals defining the CELL ARRAY elements.

NOTE that if RASTER_SIMULATE is set TRUE, then all other raster information will be ignored and the vector class POLYGON instructions will be used to display the CELL ARRAY.

Type: Logical
Default: FALSE - Do not simulate scan line instructions.
Maximum number of entries: 1

RASTER_COORD_LOWER_LEFT_X

Description: The device X coordinate of the lower left corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

RASTER_COORD_LOWER_LEFT_Y

Description: The device Y coordinate of the lower left corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

RASTER_COORD_UPPER_RIGHT_X

Description: The device X coordinate of the upper right corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

RASTER_COORD_UPPER_RIGHT_Y

Description: The device Y coordinate of the upper right corner of the drawing surface.
Type: Decimal
Default: None
Maximum number of entries: 1

RASTER_COORD_XOFFSET
Description: A device coordinate value added to X addresses before they are sent to the device. The translator attempts to center the plot based on the lower left, upper right values. When the coordinate space is asymmetric (x coord interval is not the same as the y coordinate interval) offsets will help properly place the frame. Another use is to offset the frame to a specific side of the display.

Type: Decimal
Default: 0 (zero)
Maximum number of entries: 1

**RASTER_COORD_YOFFSET**

Description: A device coordinate value added to Y addresses before they are sent to the device. See **RASTER_COORD_XOFFSET** discussion.

Type: Decimal
Default: 0 (zero)
Maximum number of entries: 1

**RASTERCOORD_XSCALE**

Description: In some cases a X unit does not equal a Y unit. This keyword allows a scale factor to be introduced into the device coordinate computation stream. It will scale the X coordinates up or down by the specified amount.

Type: Floating point
Default: 1.0
Maximum number of entries: 1

**RASTERCOORD_YSCALE**

Description: See the **RASTERCOORD_XSCALE** discussion.

Type: Floating point
Default: 1.0
Maximum number of entries: 1

**RASTER_DATA_FORMAT**

Description: The format used to organize the device raster pixel data. The input array to this format is a string of pixel color index values. Depending on the **RASTER_INSTRUCTION_START** definition the array may contain a color index entry for every pixel or it may be runlength encoded (the array contains sets of 2 values, a runlength followed by a color index). See the discussion of formatting at the beginning of this section. The example that follows describes the formatting required for the TEKTRONIX 4107 to generate a RUNLENGTH WRITE instruction. The format receives an array of runlength encoded pixel values. The first value is the pixel count the second is the color index. Each set of runlength values must be formatted such that:

\[
\text{RUNCODE} = \text{number_of_pixels} \times (2^n) + \text{color_index}
\]

Where \( n \) = the bits per pixel the device is set up to use. In our example we use 4 bits per pixel which allows the full 16 colors to be specified from the device color table. The format table will now appear as below:
Type:  Decimal rows with 4 elements per row.
Default: None
Maximum number of entries: 80 (20 rows * 4 columns)

RASTER_DATA_ENCODING
Description: The encoding scheme used for device raster data. See the Encoding discussion at the beginning of this section.
Type:  Decimal in the range of 0 to 5.
Default: 0 (zero)
Maximum number of entries: 1

RASTER_DATA_FLOATING_INFO
Description: The mapping from fixed point (integer) to output floating point. This option is used when the RASTER_DATA_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the device.
Type:  Floating point
Default: None

RASTER_VECTOR_COUNT_ENCODING
Description: The encoding scheme used for raster instruction vector counts. See the Formatting and Encoding discussion at the start of this section.
Type:  Decimal in the range of 0 to 5.
Default: 0 (zero)
Maximum number of entries: 1

RASTER_VECTOR_COUNT_FLOATING_INFO
Description: The mapping from fixed point (integer) to output format floating point. This option is used when the RASTER_VECTOR_COUNT_ENCODING is set to ascii real (5). The keyword requires 4 floating point numbers. The first two describe the minimum and maximum data values received by the encoder. The last two describe the minimum and maximum floating point values sent to the device.
Type:  Floating point
Default: None

RASTER_VECTOR_COUNT_FORMAT
Description: The format used to organize the vector count data. This formatter will receive only one input data word. See the discussion on Formatting and Encoding at the start of this section.

Example
This example encodes a 16 bit vector count.
DEVCOUNT_FORMAT /* bitstart.. DRAFT bit_count datavalue 16 1 - 15

Type: Decimal rows with 4 elements per row.
Default: None
Maximum number of entries: 40 (10 rows * 4 columns)
SECTION III: FONTCAP FORMS
This section describes the use of the FONTCAP documents. A FONTCAP document is a table describing the specifications of a character font. In its original form the document consists of a set of keywords followed by the definition of the character attributes or shapes. A keyword must be contained on one line.

After definition the CAP file must be processed by the FONTCAP processor before it can be used as input to the translator.

Definitions have four types:

1. Logical valued TRUE or FALSE. The value is stored, for the target machine, in a FORTRAN type LOGICAL.

2. Decimal valued. A positive or negative decimal value, separated by blanks. The value is stored, for the target machine, in a FORTRAN type INTEGER.

3. Floating point valued. A floating point value must contain a decimal point. Floating point values are separated by blanks. The value is stored, for the target machine, in a FORTRAN type REAL.

All keyword definitions may range from null to a maximum count specified by that keyword. Keyword definitions are allowed a maximum of 80 columns per line, however they may cross line boundaries. A new line is equivalent to a space for separating definition entities.

NOTE: keywords in the FONTCAP must appear in the order they are described in this section. There are no default definitions. All keywords must be defined unless explicitly stated.
1. CHARACTER CLASS
The following keywords define the collating sequence and the size of the characters in the font. There
may not be any holes in the collating sequence.

CHARACTER_START
Description: The decimal equivalent of the first character in the collating sequence.
Type: Decimal
Maximum number of entries: 1

CHARACTER_END
Description: The decimal equivalent of the last character in the collating sequence.
Type: Decimal
Maximum number of entries: 1

CHARACTER_HEIGHT
Description: The top of the highest character in the collating sequence. This value is given in font units.
Type: Decimal
Maximum number of entries: 1

CHARACTER_WIDTH
Description: The width of characters in a mono-space font, given in font units. A flag of -1 if this is a
variable space font.
Type: Decimal
Maximum number of entries: 1
2. FONT CLASS
This class of keywords define various character positions in the font.

FONT.CENTER
Description: The horizontal center of the font coordinate system.
Type: Decimal
Maximum number of entries: 1

FONT.RIGHT
Description: The right side of the font coordinate system.
Type: Decimal
Maximum number of entries: 1

FONT.TOP
Description: The vertical top of the font coordinate system.
Type: Decimal
Maximum number of entries: 1

FONT.CAP
Description: The vertical top of a capital letter given in the font coordinate system.
Type: Decimal
Maximum number of entries: 1

FONT.HALF
Description: The vertical center of a capital letter in the font coordinate system.
Type: Decimal
Maximum number of entries: 1

FONT.BASE
Description: The vertical base of a capital letter in the font coordinate system.
Type: Decimal
Maximum number of entries: 1

FONT.BOTTOM
Description: The vertical bottom of the font coordinate system. Extenders drop down to this level.
Type: Decimal
Maximum number of entries: 1

FONT.WIDTH
Description: The horizontal distance of the font coordinate system. If the font is variable spaced, then this is the widest character.
Type: Decimal
Maximum number of entries: 1

FONT.HEIGHT
Description: The vertical distance covered by the font coordinate system.
Type: Decimal
Maximum number of entries: 1
3. COORDINATE CLASS

The coordinate class defines where each part of a character stroke is stored in an integer word (the packing instructions). The total of the coordinate definitions must fit in an integer word on the target machine. The paint class is not used except by the NCAR DICOMED camera translator and need not be defined unless the font is to be used by the DICOMED. Paint class defines a condition of where to start the actual beam exposure for each stroke. The start positions refer to the location in an integer word to start storing a bit string. The string is stored left to right. The start position aligns with the leftmost bit of the string to be stored. The indices are valued at 0 (zero) at the right of a word and increase to the left.

COORD_X_START
Description: The starting point for storage of the X coordinate of a stroke.
Type: Decimal
Maximum number of entries: 1

COORD_Y_START
Description: The starting point for storage of the Y coordinate of a stroke.
Type: Decimal
Maximum number of entries: 1

COORD_PEN_START
Description: The starting point for storage of the pen up/down indicator.
Type: Decimal
Maximum number of entries: 1

COORD_X_LEN
Description: The number of bits in the X coordinate.
Type: Decimal
Maximum number of entries: 1

COORD_Y_LEN
Description: The number of bits in the Y coordinate.
Type: Decimal
Maximum number of entries: 1

COORD_PEN_LEN
Description: The number of bits in the pen indicator.
Type: Decimal
Maximum number of entries: 1

PAINT_BEGIN_START
Description: The starting point for storage of the paint begin indicator.
Type: Decimal
Maximum number of entries: 1

PAINT_END_START
Description: The starting point for storage of the paint end indicator.
Type: Decimal
Maximum number of entries: 1

PAINT_BEGIN_LEN
Description: The number of bits in the indicator.
Type: Decimal
Maximum number of entries: 1

PAINT_END_LEN
Description: The number of bits in the indicator.
Type: Decimal
Maximum number of entries: 1
4. CHARACTER STROKE CLASS

This class defines the characters in their proper collating sequence. It has only one keyword which delimits each character.

CHAR

Description: This keyword defines the strokes for each character in the font. Each character must be in its proper position in the collating sequence and preceded by a CHAR keyword.

A stroke consists of 5 components:

- X-stroke: The X coordinate given in font coordinate space.
- Y-stroke: The Y coordinate given in font coordinate space.
- PEN: The pen up/down bit. Zero pen up (a move) One pen down (draw).
- Paint Start bit: The draw at start of vector bit. Interpreted same as PEN. Only effective for the NCAR Dicomed translator.
- Paint End bit: The draw at end of vector bit. Interpreted same as PEN. Only effective for the NCAR Dicomed translator.

All five components must be included for each stroke. The paint bits may be Zero or One at any time unless you intend to use the font for the Dicomed.

Presented below is an example of 2 characters as defined in the FONTCAP file. The X and Y length are 5 bits. The PEN and PAINT are 1 bit.

<table>
<thead>
<tr>
<th>CHAR</th>
<th>/* */</th>
<th>x_coord</th>
<th>y_coord</th>
<th>pen</th>
<th>paint_st</th>
<th>paint_end</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>33</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>29</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAR</th>
<th>/* */ A</th>
<th>x_coord</th>
<th>y_coord</th>
<th>pen</th>
<th>paint_st</th>
<th>paint_end</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>35</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>35</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>18</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Type: Decimal

Maximum number of entries:

- For X coordinates, Y coordinates, PEN and PAINT, the user defines the range of values allowed with the COORDINATE CLASS keywords.
- There are a maximum of 60 strokes allowed per character.
- A maximum of 100 characters may be defined.
- The total stroke count cannot exceed 800.
SECTION IV: INSTALLATION REQUIREMENTS
1. Introduction

In this section the installation dependent requirements will be discussed. The installation requirements include the naming of font and graphic device definitions, metafile name passing, command line processing, and locally required routines.

All code supplied for the Graphcap processor, Fontcap processor and the Translator is written in FORTRAN 77. All character strings are restricted to be of CHARACTER*1 form. All I/O in the translator is controlled by locally written routines. The file processing I/O in the Graphcap and Fontcap processors is controlled by locally written routines. The prompts and user input in the Graphcap and Fontcap processors is controlled by FORTRAN 77 READ and WRITE statements in the respective main programs. All bit manipulation is controlled by local routines.
2. REQUIRED FILES AND COMMAND LINE PROCESSING

The files required for operation of the translator include a font definition, graphics device definition and a NCAR SPP or CGM metafile.

The font and graphics device definition must be files generated by the Fontcap and Graphcap processors respectively. The metafile must be one generated by the NCAR System Plot Package or the NCAR GKS Level 0a.

The translator needs some method of knowing the default font file, graphcap file and metafile to be used.

1. The font file name is located in the common variable FNCAPN. The variable is defined in BLOCKDATA TRNDAT. You should set this to the name of the file (full pathname) used for the default font. To define the graphics device, the variable GRCAPN must contain the name of the file that defines the instructions for the appropriate device. These files must be generated by the Fontcap and Graphcap processors.

The means to communicate the appropriate file names are specific to each installation. Presented here are examples of how to perform the communication on two popular systems. First a UNIX 4.2BSD system which has environment variables:

- Use setenv to set the environment variable FONTO to the pathname of the default font. Notice FNCAPN is defined as FONTO by default.
- The local translator routine BOPRED uses getenv on FNCAPN to get the string. If it is successful (the environment variable is defined), I open the file it points to. If it fails I attempt to open the file defined by the string in FNCAPN.

Graphcaps are opened in a similar way. Both use BOPRED to open the file.

The Metafile name is placed on the command line. The routines FNDARG and GARG are modified to parse the command line and open the name on the command line as metafile to use. If a second argument is located on the command line, it is assumed to be a record pointer into the file. This argument causes only the frame which starts at that record to be displayed. When no record pointer is included all frames in the file are displayed.

On an IBM system running VM/CMS, the default font filename is assigned to the variable FNCAPN in the BLOCKDATA TRNDAT. The device name is passed in as an argument on the command line and local command line processing is used to assign the device name to GRCAPN.

Note: command line processing for IBM VM/CMS requires modification to the main program of the Translator. To read a command line a system routine must be called by the main program. The command line is stored in a Common block and passed to FNDARG and GARG where modifications are made to parse this string and assign the appropriate names to variables. The metafile and optional record pointer are also part of the command line and work similar to the UNIX discussion above.

FNDARG is instrumented as in the UNIX discussion above. If you do not use that scheme for file name communication, you will need to modify this routine.

The strings assigned to FNCAPN and GRCAPN must be terminated by a blank. If they are terminated in another way the local routine BOPRED must have the FLNAME parameter redefined.

June 17, 1986
Page 39
NCAR CGM TRANSLATOR
3. Local Routines Required by the Translator

This part describes the routines required by the NCAR CGM Translator at each installation. All variable types are given in terms of FORTRAN 77. In the descriptions below, alloc represents no errors occurring during the execution of the described routine.

The routines GBYTES, IIMACH and SBYTES are required but not discussed here. Their description will be found in the NCAR GKS Implementation Guide.

ARGGET Get the requested argument from the command line. This is an optional routine required only if command line processing is implemented.

ARGGET (NUMBER, STRING)

This routine returns the argument requested to the caller. The exact definition of this routine is installation dependent. The code of FNDARG and GARG contain commented code using this routine as an example of how to include command line processing at your installation. At NCAR, 1 will return the path name of the metafile and 2 will return the record pointer into the metafile.

INPUT
   NUMBER - INTEGER, argument number requested.

OUTPUT
   STRING - CHARACTER*1 STRING(80), string containing the argument.

BCLRED

BCLRED (UNIT, IOS, STATUS)

Close a file opened by BOPRED for sequential binary reading.

INPUT
   UNIT - INTEGER, the unit to close

OUTPUT
   IOS - INTEGER, the I/O status, valid if STATUS is set to non-zero.
   STATUS - INTEGER, the error status.
      zero - alloc
      non-zero - error closing the unit
BINRED

**BINRED (UNIT, COUNT, BUFFER, IOS, STATUS)**

Transfer "COUNT" FORTRAN INTEGERS from an unformatted file. The file was opened by BOPRED.

**INPUT**
- UNIT - INTEGER, the unit from which to read.
- COUNT - INTEGER, the number of integer words to transfer.

**OUTPUT**
- BUFFER - INTEGER BUFFER(COUNT), the buffer of integer words read in.
- IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
- STATUS - INTEGER, the error status
  - zero - allok
  - non-zero - error reading from the unit

BOPRED

**BOPRED (UNIT, FLNAME, IOS, STATUS)**

Open a file for sequential unformatted reading. The file was created either by the GRAPHCAP or FONTCAP preprocessors using the local routine BINOPN.

**INPUT**
- UNIT - INTEGER, the unit to open.
- FLNAME - CHARACTER*1, a string containing a pathname, blank terminated.

**OUTPUT**
- UNIT - INTEGER, this value returns the file descriptor (for UNIX systems using C I/O).
- IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
- STATUS - INTEGER, the error status.
  - zero - allok
  - non-zero - error opening the unit

FLUS

**FLUS (BUFFER, COUNT, UNIT)**

Dump device-dependent instructions.

**INPUT**
- BUFFER - INTEGER, a packed array of device instructions.
- COUNT - INTEGER, the number of 8-bit bytes to transmit.
- UNIT - INTEGER, the unit where instructions are sent.
FRPRMP

**FRPRMP (BUFFER, COUNT, IOS, STATUS)**

Send the frame finished prompt and wait for response. This routine is used only if the device is declared NON-BATCH.

**INPUT**
- BUFFER - INTEGER, an array containing the user prompt. Unpacked with the low order 8 bits of every word containing an ASCII character.
- COUNT - INTEGER, the number of integer words in BUFFER.

**OUTPUT**
- IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
- STATUS - INTEGER, the error status.
  - zero - alloc
  - non-zero - error reading the user prompt

ARGC T

Note that this function is used only when optional command line processing is installed.

**INTEGER FUNCTION IARGCT (DUMMY)**

This routine returns as the function result the number of arguments on the command line. This routine will be used in conjunction with ARGGET. The exact definition of this routine is installation dependent. The code of FNDARG and GARG contain commented code using this routine as an example of how to include command line processing at your installation.

**INPUT**
- NONE - DUMMY is a dummy argument to allow FORTRAN function invocation.

**OUTPUT**
- As function result, the number of arguments on the command line at translator invocation.

MSGG

**MSGG (UNIT, MSGNUM, ADDINF)**

Put out a message on the error unit. The message should read:

```
Message Flag= MSGNUM  Other info= ADDINF
```

This corresponds to the message format given in the Status Message section of this manual.

**INPUT**
- UNIT - INTEGER, The unit number to use for writing the message.
- MSGNUM - INTEGER, The error number.
- ADDINF - INTEGER, Usually the I/O status but may be some other information.

**OUTPUT**
- NONE
MTOPEN

MTOPEN (UNIT, FLNAME, IOS, STATUS)

Open a file for direct unformatted reading of 1440 byte records. This routine opens the a
metafile written by the NCAR SPP or NCAR GKS.

INPUT

FLNAME - CHARACTER*1, a string containing the pathname of the metafile.

OUTPUT

UNIT - INTEGER, the unit number opened for reading.
IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
STATUS - INTEGER, the error status.
  zero - allok
  non-zero - error opening the file

READIT

READIT (BUFFER, MAX)

Read a character string from standard input and put it into a FORTRAN 77 CHARACTER*1 array.

INPUT

MAX - INTEGER, the maximum number of characters to read into the BUFFER.

OUTPUT

BUFFER - CHARACTER*1, array to hold the characters read from standard input.

RECRED

RECRED (UNIT, RECNUM, BUFFER, WORDS, IOS, STATUS)

Read in a metafile record using direct access I/O. The metafile is opened by the MTOPEN
routine.

INPUT

UNIT - INTEGER, the unit number from which to read.
RECNUM - INTEGER, the current metafile record to read. Count starts at 1.
  The metafile records are 1440 bytes long.
WORDS - INTEGER, The number of words to read so that 1440 bytes are
  transferred.

OUTPUT

BUFFER - INTEGER, the array containing the current metafile record.
IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
STATUS - INTEGER, the error status.
  zero - allok
  1 - error reading
  2 - end of file found
WRITIT

WRITIT (STRING, COUNT)

Write a character string to standard output.

INPUT

STRING - CHARACTER*1, array of characters to write.
COUNT - INTEGER, number of characters to write.

OUTPUT

NONE
4. Local Routines Required by FONTCAP and GRAPHCAP preprocessors

This part describes the installation dependent routines required by the FONTCAP and GRAPHCAP preprocessors. All variable types are given in terms of FORTRAN 77.

The routines GBYTES, I1MACH and SBYTES are required but not discussed here. A description of them will be found in the local routine documentation of the NCAR GKS Implementation Guide.

**BINCLS**

**BINCLS (UNIT, IOS, STATUS)**

Close a file opened for sequential binary writing. The file was opened by BINOPN.

**INPUT**

UNIT - INTEGER, the unit to close.

**OUTPUT**

IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.

STATUS - INTEGER, the error status.

zero - alloc

non-zero - error closing the unit

**BINOPN**

**BINOPN (UNIT, FLNAME, IOS, STATUS)**

Open a file for sequential binary writing.

**INPUT**

FLNAME - CHARACTER*1, string containing the path name of the file to open.

The string must be terminated with a blank.

**OUTPUT**

UNIT - INTEGER, the unit the file was opened on.

IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.

STATUS - INTEGER, the error status.

zero - alloc

non-zero - error opening the file
BINWRI

**BINWRI (UNIT, COUNT, BUFFER, IOS, STATUS)**

Transfer the contents of BUFFER to the file named by UNIT. Use a sequential unformatted write to perform the transfer. The file was opened by BINOPN.

**INPUT**

UNIT - INTEGER, the unit on which to write.
COUNT - INTEGER, the number of integers to transfer from BUFFER.
BUFFER - INTEGER, array of data which is to be transferred.

**OUTPUT**

IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero:
STATUS - INTEGER, the error status.
zero - alloc
non-zero - error writing

CHRCLS

**CHRCLS (UNIT, IOS, STATUS)**

Close a file opened for sequential character reading. The file was opened by CHROPN.

**INPUT**

UNIT - INTEGER, the unit to close.

**OUTPUT**

IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
STATUS - INTEGER, the error status.
zero - alloc
non-zero - error closing the unit

CHROPN

**CHROPN (UNIT, FLNAME, IOS, STATUS)**

Open a file for sequential unformatted reads. The file contains characters with a maximum of 80 characters per row (record).

**INPUT**

FLNAME - CHARACTER*1, string containing the path name of the file to open.
The string must be terminated with a blank.

**OUTPUT**

UNIT - INTEGER, the unit the file was opened on.
IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
STATUS - INTEGER, the error status.
zero - alloc
non-zero - error opening the file
CHRRED

CHRRED (UNIT, COUNT, BUFFER, IPTR, IOS, STATUS)

Transfer from a character file to a FORTRAN character array. The file was opened by CHROPN.

INPUT
UNIT - INTEGER, the unit from which to read.
COUNT - INTEGER, the number of characters to transfer.

OUTPUT
BUFFER - CHARACTER*1, string containing the current row of the file.
IPTR - INTEGER, points to the first valid character in BUFFER.
IOS - INTEGER, the I/O status, valid only if STATUS is set to non-zero.
STATUS - INTEGER, the error status.
  zero - alloc
  non-zero - error reading
SECTION V:
STATUS MESSAGES
This document describes the processing status of the metafile translator. This information is displayed on your device at translator termination. The messages have the form:

\[ \text{MESSAGE FLAG}=X \text{ OTHER INFO}=Y \]

The following list defines the errors generated by the translator. The message flag, X, is displayed as the list number.

1. Frame count flag. The \text{OTHER INFO} is the number of frames processed.
2. Error opening the metacode file. The \text{OTHER INFO} is a system dependent error flag indicating the reason for failure.
3. Error decoding the record position field from the command line. The record pointer given to the translator for display of a specific frame was not a valid integer.
4. Error reading a metacode record. The \text{OTHER INFO} will be a system dependent error status.
5. Metacode type change generally. This should never be displayed.
6. An invalid metacode type was detected. The translator attempts to skip and look for another record to use.
7. Invalid metacode instruction. The translator attempts to continue processing. The Element Class is displayed in Other Info.
8. The record type changed when an instruction crossed a record boundary. The translator will abort.
9. The requested frame has been plotted, not used when in display all frame mode. This should not be displayed.
10. Error encoding a integer value, typically the integer is larger than allowed by the given field length.
11. The device coordinate data type had an invalid value.
12. Error opening the graphcap file. The \text{OTHER INFO} will display a system dependent status value.
13. Error reading the graphcap file. The \text{OTHER INFO} will display a system dependent status value.
14. Error closing the graphcap file. The \text{OTHER INFO} will display a system dependent status value.
15. Error opening the fontcap file. The \text{OTHER INFO} will display a system dependent status value.
16. Error reading the fontcap file. The \text{OTHER INFO} will display a system dependent status value.
17. Error closing the fontcap file. The \text{OTHER INFO} will display a system dependent status value.
18. Polyline bundle index greater than defined. No action taken.
19. Polymarker bundle index greater than defined. No action taken.
20. Text bundle index greater than defined. No action taken.
21. Polygon bundle index greater than defined. No action taken.
22. Invalid polymarker type requested. No action taken.
23. Error in the number of cell array cells given.
24. CGM color table attribute index larger than table allows. No action taken.
25. No room for latest direct color specification. No action taken.
26. An unknown MINIMAL metacode instruction was found.
27. An unknown MINIMAL option instruction setting.
28. End of Metafile. Instruction causes the translator to terminate.
29. Metafile and Translator versions do not match. The \text{OTHER INFO} is the metafile version number.
30. Bad record size. The record bit count is not in the valid range. The OTHER INFO is the bit count of the bad record.

31. Error writing to the display surface.

32. No Version number found. A Metafile version instruction must be encountered before a Begin Picture instruction.
APPENDIX 1

This document describes the ASCII character representations used in the Graphcap. The list is broken up into two tables. Each character is represented by the code used in Graphcaps, its ASCII name, and its decimal position in the collating sequence. After the last table is a description of how to insert an arbitrary value into a byte (the INT command). Note that you will use the CHAR field representation in Graphcaps.

<table>
<thead>
<tr>
<th>CHAR</th>
<th>NAME</th>
<th>DEC</th>
<th>CHAR</th>
<th>NAME</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td>Null</td>
<td>0</td>
<td>@</td>
<td>@</td>
<td>64</td>
</tr>
<tr>
<td>SOH</td>
<td>Start of Heading</td>
<td>1</td>
<td>A</td>
<td>A</td>
<td>65</td>
</tr>
<tr>
<td>STX</td>
<td>Start of Text</td>
<td>2</td>
<td>B</td>
<td>B</td>
<td>66</td>
</tr>
<tr>
<td>ETX</td>
<td>End of Text</td>
<td>3</td>
<td>C</td>
<td>C</td>
<td>67</td>
</tr>
<tr>
<td>EOT</td>
<td>End of Transmission</td>
<td>4</td>
<td>D</td>
<td>D</td>
<td>68</td>
</tr>
<tr>
<td>ENQ</td>
<td>Enquiry</td>
<td>5</td>
<td>E</td>
<td>E</td>
<td>69</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgment</td>
<td>6</td>
<td>F</td>
<td>F</td>
<td>70</td>
</tr>
<tr>
<td>BEL</td>
<td>Bell</td>
<td>7</td>
<td>G</td>
<td>G</td>
<td>71</td>
</tr>
<tr>
<td>BS</td>
<td>Backspace</td>
<td>8</td>
<td>H</td>
<td>H</td>
<td>72</td>
</tr>
<tr>
<td>HT</td>
<td>Horizontal Tab</td>
<td>9</td>
<td>I</td>
<td>I</td>
<td>73</td>
</tr>
<tr>
<td>LF</td>
<td>Line Feed</td>
<td>10</td>
<td>J</td>
<td>J</td>
<td>74</td>
</tr>
<tr>
<td>VT</td>
<td>Vertical Tab</td>
<td>11</td>
<td>K</td>
<td>K</td>
<td>75</td>
</tr>
<tr>
<td>FF</td>
<td>Form Feed</td>
<td>12</td>
<td>L</td>
<td>L</td>
<td>76</td>
</tr>
<tr>
<td>CR</td>
<td>Carriage Return</td>
<td>13</td>
<td>M</td>
<td>M</td>
<td>77</td>
</tr>
<tr>
<td>SO</td>
<td>Shift Out</td>
<td>14</td>
<td>N</td>
<td>N</td>
<td>78</td>
</tr>
<tr>
<td>SI</td>
<td>Shift In</td>
<td>15</td>
<td>O</td>
<td>O</td>
<td>79</td>
</tr>
<tr>
<td>DLE</td>
<td>Data Link Escape</td>
<td>16</td>
<td>P</td>
<td>P</td>
<td>80</td>
</tr>
<tr>
<td>DC1</td>
<td>Device Control 1</td>
<td>17</td>
<td>Q</td>
<td>Q</td>
<td>81</td>
</tr>
<tr>
<td>DC2</td>
<td>Device Control 2</td>
<td>18</td>
<td>R</td>
<td>R</td>
<td>82</td>
</tr>
<tr>
<td>DC3</td>
<td>Device Control 3</td>
<td>19</td>
<td>S</td>
<td>S</td>
<td>83</td>
</tr>
<tr>
<td>DC4</td>
<td>Device Control 4</td>
<td>20</td>
<td>T</td>
<td>T</td>
<td>84</td>
</tr>
<tr>
<td>NAK</td>
<td>Negative Acknowledgment</td>
<td>21</td>
<td>U</td>
<td>U</td>
<td>85</td>
</tr>
<tr>
<td>SYN</td>
<td>Synchronization Character</td>
<td>22</td>
<td>V</td>
<td>V</td>
<td>86</td>
</tr>
<tr>
<td>ETB</td>
<td>End of Transmission Block</td>
<td>23</td>
<td>W</td>
<td>W</td>
<td>87</td>
</tr>
<tr>
<td>CAN</td>
<td>Cancel</td>
<td>24</td>
<td>X</td>
<td>X</td>
<td>88</td>
</tr>
<tr>
<td>EM</td>
<td>End of Medium</td>
<td>25</td>
<td>Y</td>
<td>Y</td>
<td>89</td>
</tr>
<tr>
<td>SUB</td>
<td>Substitute</td>
<td>26</td>
<td>Z</td>
<td>Z</td>
<td>90</td>
</tr>
<tr>
<td>ESC</td>
<td>Escape</td>
<td>27</td>
<td>[</td>
<td>[</td>
<td>91</td>
</tr>
<tr>
<td>FS</td>
<td>Field Separator</td>
<td>28</td>
<td>\</td>
<td>\</td>
<td>92</td>
</tr>
<tr>
<td>GS</td>
<td>Group Separator</td>
<td>29</td>
<td>]</td>
<td>]</td>
<td>93</td>
</tr>
<tr>
<td>RS</td>
<td>Record Separator</td>
<td>30</td>
<td>.</td>
<td>.</td>
<td>94</td>
</tr>
<tr>
<td>US</td>
<td>Unit Separator</td>
<td>31</td>
<td></td>
<td></td>
<td>95</td>
</tr>
</tbody>
</table>

Translator ASCII Codes Table 1

June 17, 1986
To load any 8 bit byte with a decimal value from 0 to 255 when ASCII character strings are required, use the INT prefix. If you wanted to load 4 consecutive bytes with the decimal values 12, 23, 201 and 0, use the following string:

INT12 INT23 INT201 INTO

When an instruction calls for a vector count to be included use the VC flag. For example the IMAGEN IMPress language for a Line requires the following instruction:
APPENDIX 1

CREATE_PATH
230: instruction code
vertex-count: word
[x:word y:word] vertex-count

The LINE_DRAW_INSTRUCTION_START keyword would have the following definition:

LINE_DRAW_INSTRUCTION_START

INT230 VC

This would generate the proper line drawing instruction. The vector count format would be obtained from the DEVICE_VECTOR keyword class. The LINE_DRAW_POLY_FLAG would be set TRUE. All coordinates for the line would follow the instruction and the vector count would contain the number of coordinate pairs in the instruction.

When an initial XY pair is required in the middle of a start instruction the XYC flag is used. This signals the translator to use the first coordinate pair in the instruction start and continue streaming from the second pair after the instruction start is completed. The Tektronix 4107 polygon instruction would be a good example of this feature.

BEGIN PANEL BOUNDARY
Starts a panel definition.

ESCAPE LP first_point draw_boundary

END PANEL
Terminates the panel definition.

ESCAPE LE

The POLYGON_INSTRUCTION keywords would be defined as follows.

POLYGON_INSTRUCTION_START
ESC L P XYC 0 GS

POLYGON_INSTRUCTION_TERMINATOR
ESC L E

The start instruction defines the Escape sequence followed by the first XY pair which are formatted using the DEVICE_COORDINATE information. The 0 indicates the fill should cover the panel boundary and the GS indicates that the following information will be Tektronix coordinates which will define the remainder of the polygon. Coordinates will terminate with the ESCAPE LE flag. The coordinates start from the second XY pair and are formatted using the DEVICE_COORDINATE information.

If your device requires a map address for the DEVICE_MAP_INSTRUCTION_START use the MAP string value. This reserves a place in the instruction string where a color/intensity map address will be inserted. The map address uses the color index Encoding and Formatting.

June 17, 1986
The SCD GKS Graphics Utilities

April 16, 1986

The National Center for Atmospheric Research
Scientific Computing Division

NCAR is sponsored by the National Science Foundation
The SCD GKS Graphics Utilities

This volume documents the code for all higher-level GKS Graphics Utilities. These utilities are all written in FORTRAN 77 and call GKS 0A functions to do all plotting. The utilities documented in this manual are:

AUTOGRAPH
CONRAN
CONRAS
CONRAQ
CONREC
CONRECSUPR
CONRECQCK
DASHCHAR
DASHLINE
DASHSMTH
DASHSUPR
EZMAP
GRIDAL
HASTON
HSTGRM
ISOSRF
ISOSRFIIIR
PWRITX
PWRITY
PWRZI
PWRZS
PWRZT
SRFACE
STRMLN
THREED
VELVCT

Users of the old NCAR package will notice that CONRECSMTH is no longer in the list. Since CONRECSMTH=CONREC+DASHSMTH it was decided to drop CONRECSMTH as a separate utility. WINDOW has been deleted since windowing is a part of the GKS package itself. SUPMAP has been deleted, but it is still an entry point of EZMAP. SCROLL has not yet been converted to GKS since it depends on the NSPP FLASH package which is somewhat difficult to implement using GKS 0A.
AUTOGRAPH
AUTOGRAPH
THE UNABRIDGED WRITE-UP

Author: David J. Kennison
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; copies may be obtained on request from the Publications Office of NCAR. 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.

The National Center for Atmospheric Research is operated by the University Corporation for Atmospheric Research and is sponsored by the National Science Foundation. Any opinions, 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.
## 1. INTRODUCTION

### 2. OVERVIEW

2.1 HOW TO DRAW A GRAPH ............................................. 2
2.2 A SIMPLER WAY TO DRAW A GRAPH ................................... 2
2.3 THE AUTOGRAPH CONTROL PARAMETERS ................................ 2
2.4 CONTROL PARAMETER NAMES ........................................ 3
2.5 CONTROL PARAMETERS ARE FLOATING-POINT .......................... 3
2.6 USE OF PARAMETER-ACCESS ROUTINES ................................ 4
2.7 SIDE EFFECTS OF PARAMETER-SETTING ................................ 4
2.8 SAVING AND RESTORING DEFAULTS ................................... 4
2.9 SPECIAL VALUES "NULL/1." AND "NULL/2." ......................... 4
2.10 THE GRAPH WINDOW ................................................ 5
2.11 THE GRID WINDOW .................................................. 5
2.12 THE GRID COORDINATE SYSTEM ..................................... 6
2.13 THE USER COORDINATE SYSTEM ...................................... 6
2.14 HOW TO GRAPH "X AS A FUNCTION OF Y" ............................ 7
2.15 WHAT A BACKGROUND CONSISTS OF ................................... 7
2.16 THE FOUR AXES ..................................................... 7
2.17 ABBREVIATED FORM OF AXIS-PARAMETER NAMES ................. 7
2.18 THE PARAMETER "AXIS/s/CONTROL." ................................ 8
2.19 THE PARAMETER "AXIS/s/LINE." .................................... 8
2.20 MOVING AN AXIS ..................................................... 8
2.21 THE "LABEL COORDINATE SYSTEM" ALONG AN AXIS ............... 8
2.22 POSITIONING OF MAJOR TICK MARKS ON AN AXIS .................... 9
2.23 APPEARANCE OF MAJOR TICK MARKS ................................ 10
2.24 POSITIONING/APPEARANCE OF MINOR TICK MARKS ............... 10
2.25 NUMERIC LABELS ON AN AXIS ..................................... 10
2.26 TYPES OF NUMERIC LABELS ......................................... 11
2.27 ORIENTATION OF NUMERIC LABELS .................................. 11
2.28 POSITIONING OF NUMERIC LABELS .................................. 11
2.29 CHARACTER SIZES IN NUMERIC LABELS ............................. 12
2.30 INFORMATIONAL LABELS ............................................ 12
2.31 THE PREDEFINED LABELS .......................................... 13
2.32 THE PARAMETER GROUP "LABEL." .................................. 14
2.33 THE PARAMETER GROUP "LINE." .................................... 14
2.34 ACCESSING A LABEL DEFINITION .......................... ...... 15
2.35 ACCESSING A LINE DEFINITION ...................................... 16
2.36 THE LABEL BOXES ................................................... 16
2.37 BACKGROUND OVERLAP PROBLEMS ................................... 17
2.38 DASHED-LINE PATTERNS FOR CURVES .............................. 18
2.39 THE PARAMETER GROUP "DASH." ................................... 18
2.40 PATTERNS USED BY EZY, EZXY, EZMY, AND EZMXY .............. 19
2.41 WINDOWING OF CURVES .............................................. 19
2.42 USE OF PWRTX BY AUTOGRAPH ..................................... 20
2.43 VARYING INTENSITIES, COLORS, ETC. ............................ 20
2.44 NON-STANDARD NUMERIC LABELS .................................. 21
2.45 SCATTERGRAMS AND HISTOGRAMS ................................... 21
2.46 GKS CONSIDERATIONS ............................................

3. ROUTINES .......................................................... 22
3.1 EZY (YDRA, NPTS, GLAB) ........................................ 23
3.2 EZXY (XDRA, YDRA, NPTS, GLAB) ............................. 24
3.3 EZHY (YDRA, IDXY, MANY, NPTS, GLAB) ...................... 25
3.4 EZMY (YDRA, IDXY, MANY, NPTS, GLAB) ...................... 26
3.5 ANOTAT (XLAB, YLAB, LBAC, ISET, NDSH, DSHC) ............ 27
3.6 DISPLA (LFR, LROW, LTY) ..................................... 29
3.7 AGSETP (TPGN, FUR, LURA) .................................. 30
3.8 AGSETF (TPGN, FUSR) ......................................... 32
3.9 AGSETI (TPGN, IUSR) .......................................... 33
3.10 AGSETC (TPGN, CUSR) ....................................... 33
3.11 AGSETP (TPGN, FUR, LURA) ................................ 34
3.12 AGSETF (TPGN, FUSR) ....................................... 34
3.13 AGSETI (TPGN, IUSR) ....................................... 35
3.14 AGSETC (TPGN, CUSR) ....................................... 35
3.15 AGSTUP (XDRA, NIVX, IIVX, NEVX, IIEY, ...) ............ 36
3.16 AGBACK ........................................................... 38
3.17 AGCURV (XVEC, IIEY, YVEC, NVEY, NDSH, KDSH) ......... 38
3.18 AGSAVE (INFO) .................................................. 39
3.19 AGSTR (INFO) .................................................. 40
3.20 AGBNCH (IDSH) .................................................. 40
3.21 AGDHN (IDSH) .................................................. 40
3.22 AGUOL (IAXS, FUNS, IDMA, VINP, VOPT) .................... 41
3.23 AGCHAX (IFLG, IAXS, IPRT, VILS) .......................... 41
3.24 AGCHCU (IFLG, KDSH) ....................................... 42
3.25 AGCHIL (IFLG, LNNM, LNO) .................................. 43
3.26 AGCHNL (IAXS, VILS, CHRM, NCIM, NCIM, ...) ........... 43
3.27 AGFWRT (XPOS, YPOS, CHRS, NCHS, ISIZ, IORI, ICEN) ... 44

4. PARAMETERS .......................................................... 45
4.1 'PRIMARY' ......................................................... 46
4.2 'FRAME' ......................................................... 46
4.3 'SET' ............................................................. 46
4.4 'ROW' ............................................................ 47
4.5 'INVERT' ......................................................... 48
4.6 'WINDOW' ......................................................... 48
4.7 'BACKGROUND' .................................................. 49
4.8 'NULL' ........................................................... 50
4.9 'NULL/1' ......................................................... 50
4.10 'NULL/2' ....................................................... 50
4.11 'GRAPH' ........................................................ 51
4.12 'GRAPH/LEFT' ................................................ 51
4.13 'GRAPH/RIGHT' .............................................. 51
4.14 'GRAPH/BOTTOM' .............................................. 52
4.15 'GRAPH/TOP' .................................................. 52
4.16 'GRID' .......................................................... 52
4.17 'GRID/LEFT' .................................................. 52
4.18 'GRID/RIGHT' ................................................ 53
4.19 'GRID/BOTTOM' .............................................. 53
4.20 'GRID/TOP' .................................................... 53
4.21 'GRID/SHAPE' ................................................ 53

- ii -
4.22 'X.' ................................................... 54
4.23 'X/MINIMUM.' ........................................ 55
4.24 'X/MAXIMUM.' ........................................ 55
4.25 'X/LOGARITHMIC.' .................................. 55
4.26 'X/ORDER.' ........................................... 56
4.27 'X/NICE.' ............................................. 56
4.28 'X/SMALLEST.' ....................................... 56
4.29 'X/LARGEST.' ......................................... 57
4.30 'Y.' ................................................... 57
4.31 'Y/MINIMUM.' ......................................... 57
4.32 'Y/AX.' ................................................ 57
4.33 'Y/LOGARITHMIC.' .................................... 58
4.34 'Y/ORDER.' ............................................ 58
4.35 'Y/NICE.' ............................................. 58
4.36 'Y/SMALLEST.' ........................................ 58
4.37 'Y/LARGEST.' .......................................... 59
4.38 'AXIS.' ............................................... 59
4.39 'AXIS/'s' .............................................. 59
4.40 'AXIS/'s/CONTROL.' .................................... 60
4.41 'AXIS/'s/LINE.' ....................................... 60
4.42 'AXIS/'s/INTERSECTION.' ............................ 60
4.43 'AXIS/'s/INTERSECTION/GRID.' ........................ 61
4.44 'AXIS/'s/INTERSECTION/USER.' ........................ 61
4.45 'AXIS/'s/FUNCTION.' .................................. 61
4.46 'AXIS/'s/TICKS.' ..................................... 62
4.47 'AXIS/'s/[TICKS]/MAJOR.' ............................ 62
4.48 'AXIS/'s/[TICKS]/MAJOR/SPACING.' .................. 62
4.49 'AXIS/'s/[TICKS]/MAJOR/[SPACING]/TYPE.' ........... 63
4.50 'AXIS/'s/[TICKS]/MAJOR/[SPACING]/BASE.' .......... 63
4.51 'AXIS/'s/[TICKS]/MAJOR/[SPACING]/COUNT.' ........ 64
4.52 'AXIS/'s/[TICKS]/MAJOR/PATTERN.' .................. 64
4.53 'AXIS/'s/[TICKS]/MAJOR/LENGTH.' .................... 64
4.54 'AXIS/'s/[TICKS]/MAJOR/[LENGTH]/OUTWARD.' ....... 65
4.55 'AXIS/'s/[TICKS]/MAJOR/[LENGTH]/INWARD.' ....... 65
4.56 'AXIS/'s/[TICKS]/MINOR.' ............................ 66
4.57 'AXIS/'s/[TICKS]/MINOR/SPACING.' ................. 66
4.58 'AXIS/'s/[TICKS]/MINOR/PATTERN.' .................. 66
4.59 'AXIS/'s/[TICKS]/MINOR/LENGTH.' ................... 67
4.60 'AXIS/'s/[TICKS]/MINOR/[LENGTH]/OUTWARD.' ...... 67
4.61 'AXIS/'s/[TICKS]/MINOR/[LENGTH]/INWARD.' ....... 67
4.62 'AXIS/'s/[NUMERIC/]' ............................... 67
4.63 'AXIS/'s/[NUMERIC/]/TYPE.' ........................ 68
4.64 'AXIS/'s/[NUMERIC/]/EXPONENT.' .................... 72
4.65 'AXIS/'s/[NUMERIC/]/FRACTION.' .................... 72
4.66 'AXIS/'s/[NUMERIC/]/ANGLE.' ........................ 72
4.67 'AXIS/'s/[NUMERIC/]/ANGLE/1ST.' .................. 73
4.68 'AXIS/'s/[NUMERIC/]/ANGLE/2ND.' .................. 73
4.69 'AXIS/'s/[NUMERIC/]/OFFSET.' ........................ 73
4.70 'AXIS/'s/[NUMERIC/]/WIDTH.' ........................ 74
4.71 'AXIS/'s/[NUMERIC/]/WIDTH/MANTISSA.' ............. 74
4.72 'AXIS/'s/[NUMERIC/]/WIDTH/EXponent.' ............. 74
4.73 'DASH.' ............................................... 74
4.74 'DASH/SELECTOR.' .................................... 75
4.75 'DASH/LENGTH.' ........................................ 75
4.76 'DASH/CHARACTER.' ..................................... 76
4.77 'DASH/DOLLAR-QUOTE.' .................................. 76
4.78 'DASH/PATTERNS.' ........................................ 76
4.79 'DASH/PATTERNS/n' ....................................... 77
4.80 'LABEL.' .................................................. 77
4.81 'LABEL/CONTROL.' ......................................... 77
4.82 'LABEL/BUFFER.' .......................................... 78
4.83 'LABEL/BUFFER/LENGTH.' ................................ 78
4.84 'LABEL/BUFFER/CONTENTS.' .............................. 78
4.85 'LABEL/BUFFER/NAMES.' .................................. 80
4.86 'LABEL/NAME.' ............................................. 80
4.87 'LABEL/DEFINITION.' ....................................... 81
4.88 'LABEL/[DEFINITION/] SUPPRESSION.' .................. 81
4.89 'LABEL/[DEFINITION/] BASEPOINT.' ..................... 81
4.90 'LABEL/[DEFINITION/] BASEPOINT/X.' ................. 82
4.91 'LABEL/[DEFINITION/] BASEPOINT/Y.' ................. 82
4.92 'LABEL/[DEFINITION/] Offset.' .......................... 82
4.93 'LABEL/[DEFINITION/] OFFSET/X.' ....................... 82
4.94 'LABEL/[DEFINITION/] OFFSET/Y.' ...................... 83
4.95 'LABEL/[DEFINITION/] ANGLE.' .......................... 83
4.96 'LABEL/[DEFINITION/] CENTERING.' ..................... 83
4.97 'LABEL/[DEFINITION/] LINES.' .......................... 83
4.98 'LABEL/[DEFINITION/] INDEX.' .......................... 84
4.99 'LINE.' .................................................. 84
4.100 'LINE/MAXIMUM.' ........................................ 84
4.101 'LINE/3-D.' ............................................. 85
4.102 'LINE/BUFFER.' .......................................... 85
4.103 'LINE/BUFFER/LENGTH.' ................................ 85
4.104 'LINE/BUFFER/CONTENTS.' ............................... 86
4.105 'LINE/NUMBER.' ......................................... 87
4.106 'LINE/DEFINITION.' ...................................... 87
4.107 'LINE/[DEFINITION/] SUPPRESSION.' .................. 88
4.108 'LINE/[DEFINITION/] CHARACTER.' ..................... 88
4.109 'LINE/[DEFINITION/] TEXT.' ............................ 88
4.110 'LINE/[DEFINITION/] LENGTH.' ........................ 89
4.111 'LINE/[DEFINITION/] INDEX.' ........................... 89
4.112 'SECONDARY.' ........................................... 89
4.113 'SECONDARY/GRAPH.' .................................... 89
4.114 'SECONDARY/USER.' ...................................... 90
4.115 'SECONDARY/CURVE.' .................................... 90
4.116 'SECONDARY/DIMENSIONS.' ............................... 90
4.117 'SECONDARY/AXIS.' ...................................... 91
4.118 'SECONDARY/AXIS/s.' .................................... 91
4.119 'SECONDARY/AXIS/s/POSITION.' ......................... 91
4.120 'SECONDARY/AXIS/s/TICKS.' ............................. 91
4.121 'SECONDARY/AXIS/s/NUMERIC.' .......................... 92
4.122 'SECONDARY/LABEL.' .................................... 92
4.123 'SECONDARY/LABEL/b.' .................................. 93

5. EXAMPLES.................................................. 95
5.1 EXAMPLE 1................................................ 96
5.2 EXAMPLE 2................................................ 98
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>EXAMPLE 3</td>
<td>100</td>
</tr>
<tr>
<td>5.4</td>
<td>EXAMPLE 4</td>
<td>102</td>
</tr>
<tr>
<td>5.5</td>
<td>EXAMPLE 5</td>
<td>104</td>
</tr>
<tr>
<td>5.6</td>
<td>EXAMPLE 6</td>
<td>107</td>
</tr>
<tr>
<td>5.7</td>
<td>EXAMPLE 7</td>
<td>111</td>
</tr>
<tr>
<td>5.8</td>
<td>EXAMPLE 8</td>
<td>116</td>
</tr>
<tr>
<td>5.9</td>
<td>EXAMPLE 9</td>
<td>119</td>
</tr>
<tr>
<td>5.10</td>
<td>EXAMPLE 10</td>
<td>124</td>
</tr>
<tr>
<td>5.11</td>
<td>EXAMPLE 11</td>
<td>130</td>
</tr>
<tr>
<td>5.12</td>
<td>EXAMPLE 12</td>
<td>132</td>
</tr>
<tr>
<td>5.13</td>
<td>A FINAL EXAMPLE</td>
<td>135</td>
</tr>
<tr>
<td>6.</td>
<td>MESSAGES</td>
<td>138</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

This is the unabridged write-up of AUTOGRAPH - a graphics package enabling the user to draw graphs, each with a labelled background and each displaying one or more curves.

This write-up is divided into several major sections: The section "OVERVIEW" presents the philosophy of the package and should eventually be read by any serious user of AUTOGRAPH. The section "ROUTINES" describes the various routines in the package and how to call them. The section "PARAMETERS" describes the "control parameters" which govern the behavior of AUTOGRAPH. The section "EXAMPLES" contains a set of graphs produced by AUTOGRAPH and the programs which produced them. The section "MESSAGES" describes the messages that accompany the possible error exits.

Each major section is divided into titled paragraphs, which are further divided into untitled paragraphs. In general, phrases like "the preceding paragraph" or "the following paragraph" are to be interpreted to refer to a titled paragraph.

Readers who wish only to quickly draw a simple graph may want to skip to the descriptions of the routines EZY, EZXY, EZMY, and EZMXY, in the section "ROUTINES". Others may wish to look first at the examples.
2. OVERVIEW

This section describes the philosophy of AUTOGRAPH.

2.1 HOW TO DRAW A GRAPH. To draw a graph, a user program executes, directly or indirectly, a series of calls to AUTOGRAPH routines, typically as follows:

1. The routines AGSETP, AGSETF, AGSETI, and/or AGSETC are called to reset "primary control parameters" having unsatisfactory values.

2. The routine AGSTUP is called to perform "set-up" tasks (thus the name). It computes appropriate values for the "secondary control parameters".

3. The routine AGRBACK is called to draw a background.

4. The routine AGCURV is called one or more times (once per curve) to draw the desired curves.

5. The system-plot-package routine FRAME is called to advance to a new frame.

To draw the next graph, all five steps are repeated; step 1 may, of course, be abbreviated or omitted entirely.

2.2 A SIMPLER WAY TO DRAW A GRAPH. Each of the routines EZY, EZXY, EZMY, and EZMXY performs a sequence of calls like that described above. A user program may call EZY to graph a single curve defined by the points (I,Y(I)), for I from 1 to N, EZXY to graph a single curve defined by the points (X(I),Y(I)), for I from 1 to N, EZMY to graph the M curves defined by the points (I,Y(I,J)), for I from 1 to N and J from 1 to M, and EZMXY to graph the M curves defined by the points (X(I),Y(I,J)) or (X(I,J),Y(I,J)), for I from 1 to N and J from 1 to M.

See the descriptions of these routines, in the section "ROUTINES". See also examples 1 through 4, in the section "EXAMPLES".

2.3 THE AUTOGRAPH CONTROL PARAMETERS. The labeled common block AGCONP contains the AUTOGRAPH "control parameters", each of which controls some aspect of the package's behavior. There are two types of these: "primary control parameters" and "secondary control parameters".

Each primary control parameter has a default value and is subject to change by a user program to produce a desired effect.

Each secondary control parameter is computed by AUTOGRAPH itself and is not normally subject to direct change by a user program. The values computed for some of the secondary control parameters may be of interest.

Access to all of the control parameters is provided by the routines AGSETP, AGSETF, AGSETI, AGSETC, AGGETP, AGGETF, AGGETI, and AGGETC. (The routines ANOTAT and DISPLA provide access to a limited subset of the control parameters and are provided principally for historical reasons; they are of interest mainly to users of the routines EZY, EZXY, EZMY, and EZMXY.)
In the following discussion, the long phrase "primary control parameter" will usually be shortened to just "parameter".

2.4 CONTROL PARAMETER NAMES. There are many groups of parameters. Each group has a keyword associated with it - like BACKGROUND or GRAPH or AXIS. Those groups which contain more than one parameter are divided into subgroups, each of which also has a keyword associated with it. The subgroups may be further subdivided in the same manner.

Group keywords are used to make up names of parameter groups and, ultimately, of individual parameters. This is done by stringing together the group keywords, in descending order, separated by slashes and terminated by a period. For example, the name "AXIS." refers to a group of 92 parameters describing the four axes, the name "AXIS/LEFT." to a subgroup of 23 parameters describing the left axis, the name "AXIS/LEFT NUMERIC." to a further subgroup of 8 parameters describing the numeric labels on the left axis, and the name "AXIS/LEFT NUMERIC/TYPEx." to a single parameter describing the type of numeric labels on the left axis.

Parameter-group names are used as arguments in calls to parameter-access routines to identify the parameter(s) whose values a user program wishes to "set" or "get". For example, the statement

```
CALL AGSETF ('AXIS/LEFT/NUMERIC/TYX.',1.)
```

is used to set the value of the parameter specifying the type of numeric labels to be used on the left axis to "1."

Parameter-group names may be shortened considerably, both by abbreviation of the keywords and by omission of some keywords; for example, the name shown above may be shortened to 'LE/TY.X.'.

Complete information about the control parameters and their names is given below, in the section "PARAMETERS".

2.5 CONTROL PARAMETERS ARE FLOATING-POINT. All of the control parameters are floating-point - even those which serve as type specifiers, control flags, item counts, list pointers, and the like - for which integer variables would normally be used. This was done because of a portability problem which arose in implementing the parameter-access routines.

Those parameters which may only have discrete integral values are referenced internally using the FORTRAN function IFIX. For example: The parameter "X/NICE." corresponds to a variable in the common block AGCONP named QCEX, which may have any of the values "-1.", "0.", or "+1.". The function IFIX(QCEX) is used by AUTOGRAPH to recover the integer value.

Conceptually, some parameters have character-string values; for example, the parameter "DASH/PATTERN/1." may, conceptually, have the value "$$$$$$1$$$$$$". Obviously, one must come up with a scheme which will allow any possible character string to be represented as a floating-point number. At one time, the floating-point equivalent of the memory address of the character string was used as the actual value of the parameter. This approach led to portability...
problems and has been abandoned in favor of the following: A character string which is to become the conceptual value of a parameter is stashed in a character-string array inside AUTOGRAPH and a floating-point identifier which will enable later retrieval of the string is stored as the actual value of the parameter. As it happens, such identifiers are always negative; positive values may therefore have other uses. For example, if "DASH/PATTERN/1." has a negative value, a character-string dash pattern is implied, but, if it has a positive value, a 16-bit binary dash pattern is implied.

2.6 USE OF PARAMETER-ACCESS Routines. The routine AGSETP (AGGETP) is called by a user program to "set" ("get") the floating-point values of a specified group of related parameters.

The routine AGSETP is used to store a floating-point number as the value of a single specified parameter, the routine AGGETP to retrieve the floating-point value of a single specified parameter.

The routine AGSETI is used to store the floating-point equivalent of an integer as the value of a single specified parameter, the routine AGGETI to retrieve the integer equivalent of the value of a single specified parameter.

The routine AGSETC is used to store a character string as the (conceptual) value of a single specified parameter, the routine AGGETC to retrieve the (conceptual) character-string value of a single specified parameter.

2.7 SIDE EFFECTS OF PARAMETER-SETTING. Setting certain individual parameters results, as a side effect, in "special action" by the routine AGSETP. For example, when "BACKGROUND." is given a new value, other parameters are also changed to create the desired background. These side effects occur whether AGSETP is reached directly from the user program or indirectly, by way of a call to one of the routines AGSETP, AGSETI, or AGSETC. They do not occur when AGSETP is asked to set one of the parameters in question as part of a multi-parameter group, only when it is asked to set that parameter individually.

2.8 SAVING AND RESTORING DEFAULTS. Parameters whose values have been changed by the user do not automatically revert to their default values. Re-creating the default state of AUTOGRAPH by resetting individual parameters can become quite tedious. The routines AGSAVE, which saves the current state of AUTOGRAPH on a file, and AGRSTR, which restores a saved state of AUTOGRAPH from a file, should be used instead. These routines are described in the section "ROUTINES".

2.9 SPECIAL VALUES 'NULL/1.' AND 'NULL/2.'. The parameters 'NULL/1.' and 'NULL/2.' define the special values "null 1" and "null 2", which have the default values "1.E36" and "2.E36", respectively. These special values are used in a couple of ways:

- Certain parameters may be given an "actual" value or one of the "null" values. An actual value directly expresses the value of some quantity. The value "null 1" specifies that AUTOGRAPH is to choose an appropriate value to use, but that it is not to store that value in place of the "null 1". The value "null 2" specifies that AUTOGRAPH is to choose an appropriate actual value and store that actual value in place of the
Example: "Y/MINIMUM.", which specifies the minimum y coordinate, has the default value "null 1", specifying that, for each graph, AUTOGRAPH is to choose (by examining the data) an appropriate minimum y. This parameter may be given an actual (non-null) value, thus imposing a desired minimum y, or it may be given the value "null 2", specifying that AUTOGRAPH is to choose an appropriate minimum y for the next graph and then use that value for following graphs.

- The value "null 1" is used in x/y coordinate data to signal missing points.

Note: If your x/y coordinate data might include the values "1.E36" or "2.E36", your program's first action should be to change the values of "NULL/1." and "NULL/2." to values which cannot possibly occur in the data.

2.10 THE GRAPH WINDOW. The parameters "GRAPH/LEFT.", "GRAPH/RIGHT.", "GRAPH/BOTTOM.", and "GRAPH/TOP." serve to locate the edges of a rectangular "graph window" within the plotter frame. The first two are stated as fractions of the frame width, the second two as fractions of the frame height. These parameters have the default values "0.", "1.", "0.", and "1.", respectively, specifying a graph window which fills the entire plotter frame.

The graph window is the area in which a graph, including labels, is to be drawn. A user program may limit a graph to any selected portion of the plotter frame. For example, changing the values of the parameters in the group "GRAPH." to "0.", "5.", "0.", and "5.", limits a graph to the lower left-hand quarter of the frame.

See example 6, in the section "EXAMPLES".

2.11 THE GRID WINDOW. The parameters "GRID/LEFT.", "GRID/RIGHT.", "GRID/BOTTOM.", and "GRID/TOP." serve to locate the edges of a rectangular area within the graph window; the parameter "GRID/SHAPE." specifies the shape of a "grid window", to be centered in, and made as large as possible in, this area. The positions of the left and right edges are stated as fractions of the graph-window width and have default values ".15" and ".95"; the positions of the bottom and top edges are stated as fractions of the graph-window height and also have default values ".15" and ".95". The parameter "GRID/SHAPE." has the default value "0.", specifying a grid window which completely fills the area specified by the other parameters. Other values allow one to specify a grid window of any desired rectangular shape or of a shape implied by the x and y coordinate data.

The grid window is the portion of the graph window along the edges of which the axes are to be drawn and within which curves are to be drawn. Numeric and informational labels are ordinarily placed in the portion of the graph window which is outside the grid window. Various positioning parameters are stated in a "grid coordinate system" based on the grid window and curve-point coordinates are given by the user in a "user coordinate system" which maps into the grid window. If "WINDOW." has the value "1.", only those curve portions which lie inside the grid window are drawn. Character sizes and label-offset
distances are specified as fractions of the smaller dimension of the grid window, so as to be in scale with the rest of the graph.

See example 7, in the section "EXAMPLES".

2.12 **THE GRID COORDINATE SYSTEM.** Internally, AUTOGRAPH makes use of a "grid coordinate system"; the user also makes use of this system at times in setting certain parameter values. The origin of the grid coordinate system is at the lower left-hand corner of the grid window. X coordinates run linearly from "0." to "1." horizontally, and y coordinates linearly from "0." to "1." vertically, in the grid window. Note that coordinate values outside the range (0.,1.) may be used to reference points outside the grid window.

2.13 **THE USER COORDINATE SYSTEM.** Curve-defining points are stated by a user program in the "user coordinate system". Fourteen parameters specify how that user coordinate system is mapped into the grid window. (The parameter "INVERT.", described below, might be considered a fifteenth.)

The first seven of the fourteen, named "X/MINIMUM.", "X/MAXIMUM.", "X/LOGARITHMIC.", "X/ORDER.", "X/NICE.", "X/SMALLEST.", and "X/LARGEST.", specify how user x coordinates are to be mapped onto the horizontal axis of the grid window. The default values of these parameters are such that the routine AGSTUP is forced to:

- Compute, from the user's x-coordinate data, minimum and maximum values X\textsubscript{m} and X\textsubscript{M}.
- Compute "nice" (rounded) values X\textsubscript{m}' and X\textsubscript{M}' such that the interval (X\textsubscript{m}, X\textsubscript{M}) is completely contained in the interval (X\textsubscript{m}', X\textsubscript{M}').
- Map X\textsubscript{m}' to the left edge, and X\textsubscript{M}' to the right edge, of the grid window. The mapping is linear.

The other seven parameters, named "Y/MINIMUM.", "Y/MAXIMUM.", etc., specify how user y coordinates are to be mapped onto the vertical axis of the grid window. The default values specify a mapping analogous to that of x coordinates.

By changing the values of these fourteen parameters appropriately, a variety of desirable ends may be achieved:

- Values of X\textsubscript{m}, X\textsubscript{M}, Y\textsubscript{m}, and/or Y\textsubscript{M} may be specified, thus limiting the graph to a particular range of interest and/or forcing consistent scaling of a group of graphs. If the selection of these values is left up to AUTOGRAPH, a range of acceptable values may be specified.
- Either or both mappings may be made logarithmic. (The logarithms of coordinate values are mapped linearly onto the axis.)
- Either or both mappings may be flipped end-for-end. X coordinates may be made to decrease from left to right, y coordinates to decrease from bottom to top.
2.14 HOW TO GRAPH "X AS A FUNCTION OF Y". The parameter 'INVERT.' has the default value "0.". If it is set to "1." by a user program, the routines AGSTUP and AGCURV will behave as if their x and y arguments had been interchanged. In some sense, this provides a way of plotting "x as a function of y".

This parameter is of principal interest to the users of EZY, EZXY, EZMY, and EZMXY; those users who call the routines AGSTUP and AGCURV directly should probably leave the parameter zeroed.

See example 8, in the section "EXAMPLES".

2.15 WHAT A BACKGROUND CONSISTS OF. A background drawn by the routine AGRACK consists of four axes and up to eight informational labels, each of the latter having none or more lines of text; the total number of such lines must not exceed sixteen. Each of these entities is defined by a group of parameters and may be modified in a variety of ways.

2.16 THE FOUR AXES. The four axes are positioned along the edges of the grid window. There are a left y axis, a right y axis, a bottom x axis, and a top x axis. Each of the axes consists of a line, major tick marks, minor tick marks, and numeric labels. Numeric labels are placed at major-tick-mark positions.

The axes are defined by the parameter group named 'AXIS.', which has subgroups 'AXIS/LEFT.', 'AXIS/RIGHT.', 'AXIS/BOTTOM.', and 'AXIS/TOP.'. Each of these subgroups contains 23 parameters defining one of the four axes. These 23 parameters fall into six further subgroups, having the associated keywords CONTROL, LINE, INTERSECTION, FUNCTION, TICKS, and NUMERIC.

The default values of the axis parameters specify a "perimeter" background: all four axes are drawn; each has short, inward-pointing major and minor ticks; the left axis and the bottom axis have numeric labels (placed outside the grid window); the right-axis and top-axis numeric labels are suppressed. See examples 1 through 4, in the section "EXAMPLES".

A "half-axis" background is created by suppressing the right axis and the top axis completely. A "grid" background is created by extending the left-axis and bottom-axis ticks all the way across the grid window and suppressing the ticks on the other two axes. The parameter 'BACKGROUND.' allows the user to create these standard backgrounds easily; whenever its value is changed by a user-program call to AGSETP, AGSETF, or AGSETT, parameters in the group "AXIS." are modified to create the desired background. See examples 5, 6, and 8, in the section "EXAMPLES".

2.17 ABBREVIATED FORM OF AXIS-PARAMETER NAMES. In the ensuing discussions of the various parameters in the group "AXIS.", the character "s" is used to stand for any one of the keywords "LEFT", "RIGHT", "BOTTOM", or "TOP". For
example, "AXIS/s/LINE." stands for any one of "AXIS/LEFT/LINE."
"AXIS/RIGHT/LINE.", etc. This form is shorter and makes it clear that four
different parameters or groups of parameters are being described at once.

2.18 THE PARAMETER "AXIS/s/CONTROL." This parameter may be given any
integral value from "-1." to "+4."

* The value "-1." specifies that only the line portion of the axis speci-
  fied by "s" is to be drawn.

* The value "0." specifies that no portion of the axis is to be drawn.

* A value from "1." to "4." specifies that all portions of the axis are to
  be drawn and tells AUTOGRAPH what liberties it may take in attempting to
  cope with numeric labels which will not fit along the axis without over-
  lapping.

The precise meanings of each value are described in the section "PARAMETERS".

2.19 THE PARAMETER "AXIS/s/LINE." This parameter has the default value
"0.". Setting it to a "1." causes the line portion of the axis specified by
"s" to be suppressed. Tick marks and/or numeric labels may still be drawn.

2.20 MOVING AN AXIS. Both of the parameters

  "AXIS/s/INTERSECTION/GRID."
  "AXIS/s/INTERSECTION/USER."

have the default value "null 1", specifying that the axis "s" is to be drawn
in its normal position, along the edge of the grid window. If either param-
eter is given a non-null value, the axis "s" is moved away from its normal
position in such a way as to intersect the sides of the grid which are perpen-
dicular to it at a point specified by that non-null value. To move an x axis,
a y coordinate is specified; to move a y axis, an x coordinate is specified.

The coordinate may be specified in the grid coordinate system or in the user
coordinate system, depending on which parameter is used. If both parameters
are given non-null values, the user-system value takes precedence.

No axis may be moved outside the graph window. Attempting to do so moves the
axis as far as the edge, but no farther.

See example 8, in the section "EXAMPLES".

2.21 THE "LABEL COORDINATE SYSTEM" ALONG AN AXIS. Each of the four axes has
associated with it a "user" coordinate system (as described above, in the
paragraph "THE USER COORDINATE SYSTEM") and a "label" coordinate system. The
routine AGUITOL defines the relationship between the two coordinate systems for
each of the four axes.

Tick marks are positioned at "nice" values in the label coordinate system,
mapped to the user coordinate system, and then mapped onto the axis. Numeric
labels are associated with major ticks and provide values in the label
coordinate system.

The default version of AGUTOL defines the label system on each axis to be identical with the user system; a private version may be substituted in place of the default in order to change the label coordinate system for any one or more of the four axes.

Example: Suppose that the y-coordinate data is in miles/hour and it is desired that the left y axis be tick-marked and labeled in meters/second. The user program must include a subroutine AGUTOL with five arguments (four input, one output), as follows:

1. the number of the axis being drawn (1, 2, 3, or 4, implying the left, right, bottom, and top axes, respectively).
2. the value of 'AXIS/s/FUNCTION.' for that axis.
3. an integer specifying whether to map from the user system to the label system (+1) or vice-versa (-1).
4. an input value in one coordinate system.
5. an output value in the other system.

The user routine must, for the left axis only, multiply the input value by the appropriate constant and return the result as the output value; for all other axes, it must return an output value equal to the input value.

It is recommended that the default value of 'AXIS/s/FUNCTION.' (zero) be used to imply that AGUTOL should do the identity mapping for the axis "s"; other values may be used to select desired mappings. This gives a way to "turn off" the use of a special mapping for a given axis.

Note that the tick-marking and labeling of one x (y) axis of a graph may be completely different from that of the other x (y) axis of the graph. For example, the left y axis could be made to indicate "height in kilometers" and the right y axis "pressure in millibars".

See example 7, in the section "EXAMPLES".

2.22 POSITIONING OF MAJOR TICK MARKS ON AN AXIS. The parameter group named 'AXIS/s/TICKS/MAJOR/SPACING.' contains three parameters, with associated keywords TYPE, BASE, and COUNT. These parameters are described in detail in the section "PARAMETERS". Major tick marks may be spaced linearly or logarithmically in the label coordinate system along the axis specified by "s", or suppressed altogether. Each of the TYPE and BASE parameters has the default value "null 1", allowing AUTOGRAPH to position major tick marks as it sees fit.

See examples 7 and 8 and the final example, in the section "EXAMPLES".
2.23 APPEARANCE OF MAJOR TICK MARKS. The parameter

'AXIS/s/TICKS/MAJOR/PATTERN.'

has an integral value from "0." to "65535." and specifies the dashed-line pattern to be used for major ticks on the axis specified by "s". Each "0" bit in the lower 16 bits of the integral value specifies a gap 3 plotter units long, each "1" bit a solid portion 3 plotter units long. The default value "65535." (2 to the 16th minus 1) specifies a solid line. The value "0." may be used to suppress major tick marks on the axis "s".

The parameters

'AXIS/s/TICKS/MAJOR/LENGTH/OUTWARD.'
'AXIS/s/TICKS/MAJOR/LENGTH/INWARD.'

specify the lengths of the outward-pointing and inward-pointing portions of the major ticks. Each is stated as a fraction of the smaller dimension of the grid window. If either of these values is made greater than or equal to "1.", it specifies a tick-mark portion which extends to the edge of the grid window and a little beyond, the magnitude of the "little beyond" being specified by the fractional portion of the parameter value. The default values give inward-pointing major ticks of length ".015" on all axes.

See example 8, in the section "EXAMPLES".

2.24 POSITIONING/APPEARANCE OF MINOR TICK MARKS. The parameter

'AXIS/s/TICKS/MINOR/SPACING.'

specifies the number of minor ticks which are to occur between pairs of major ticks on the axis specified by "s". Minor ticks are equidistantly spaced in the label coordinate system for that axis. The default value of this parameter is "null 1", allowing AUTOGRAPH to position minor ticks as it sees fit.

The parameters

'AXIS/s/TICKS/MINOR/PATTERN.',
'AXIS/s/TICKS/MINOR/LENGTH/OUTWARD.'
'AXIS/s/TICKS/MINOR/LENGTH/INWARD.'

specify the dashed-line pattern, outward-pointing length, and inward-pointing length of minor ticks. They are defined in the same way as the analogous major-tick parameters, except that the default inward-pointing tick length is ".010".

See the final example, in the section "EXAMPLES".

2.25 NUMERIC LABELS ON AN AXIS. The parameter group named 'AXIS/s/NUMERIC.' contains eight parameters describing the numeric labels on the axis specified by "s". These parameters are described in detail in the section "PARAMETERS"; they are described sketchily in succeeding paragraphs.
2.26 TYPES OF NUMERIC LABELS. The parameter named 'AXIS/s/NUMERIC/TYPE.' may be given any integral value from "0." to "3." or one of the values "null 1" or "null 2".

- The value "0." suppresses numeric labels on the axis specified by "s".
- The values "1.", "2.", and "3." specify the use of "scientific notation", "exponential notation", and "no-exponent notation", respectively.
- A null value gives AUTOGRAPH the freedom to use one of the values "1.", "2.", or "3."—whichever is most consistent with the label coordinate system along the axis.

The exact nature of the labels produced by a given value depends on the three parameters

'AXIS/s/TICKS/MAJOR/SPACING/TYPE.'
'AXIS/s/NUMERIC/EXponent.'
'AXIS/s/NUMERIC/FRACTION.'

See example 7 and the final example, in the section "EXAMPLES".

2.27 ORIENTATION OF NUMERIC LABELS. The parameters

'AXIS/s/NUMERIC/ANGLE/1ST.'
'AXIS/s/NUMERIC/ANGLE/2ND.'

may have integral values "0.", "90.", "180.", or "270.". They specify the user's first and second choices for the orientation of numeric labels on the axis specified by "s". AUTOGRAPH will attempt to use the first choice (default value — "0." for all axes); if that leads to overlap problems and shrinking the labels either doesn't help or is not permitted and rotation is permitted (by the setting of 'AXIS/s/CONTROL.'), AUTOGRAPH may try the second choice (default value — "90." for all axes).

The values given represent angles measured in degrees counter-clockwise from horizontal.

2.28 POSITIONING OF NUMERIC LABELS. The parameter

'AXIS/s/NUMERIC/OFFSET.'

specifies on which side of the axis specified by "s" the numeric labels are to lie and the size of the gap to be left between the axis line and the numeric labels.

- A negative value specifies labels inside the grid window.
- A zero value specifies labels centered on the axis, suppresses the line portion of the axis, and moves the inward-pointing and outward-pointing portions of ticks out away from the axis so as to leave room for the labels.
A positive value specifies labels outside the grid window.

The magnitude of the value specifies the distance from the axis to the nearest portion of the label, stated as a fraction of the smaller side of the grid window.

The default value for all axes is ".015".

2.29 CHARACTER SIZES IN NUMERIC LABELS. The parameters

'AXIS/s/NUMERIC/WIDTH/MANTISSA.'
'AXIS/s/NUMERIC/WIDTH/EXponent.'

specify the widths of characters in the mantissa and exponent portions of the numeric labels on the axis specified by "s", stated as fractions of the smaller dimension of the grid window.

The sizes specified are those desired by the user. If an overlap problem arises and "AXIS/s/CONTROL." is set so as to allow AUTOGRAPH to shrink the numeric labels, the characters may end up smaller than desired. No character is shrunk to less than the minimum readable size, however.

These parameters have default values ".015" and ".010", respectively, for all axes.

2.30 INFORMATIONAL LABELS. As many as m informational labels may be defined at any one time; normally, m = 8. The informational labels form a part of the background produced by a call to the routine AGBACK. Each of the informational labels is defined as follows:

- Each label has a name - a short character string which uniquely identifies the label.

- Each label has a "suppression flag", which may be set to enable or disable plotting of the label.

- Each label is positioned relative to a "basepoint", whose x and y coordinates are specified in the grid coordinate system. Normally, the basepoint lies on one edge - but not at a corner point - of the grid window.

- Emanating from the label basepoint is an "offset vector", whose x and y components are specified as signed fractions of the smaller dimension of the grid window. Normally, the offset vector is used to specify the size of the gap to be left between an informational label and the edge of the grid window. The presence or absence of an axis along that edge of the grid window is ignored when specifying this gap; see the paragraph "BACKGROUND OVERLAP PROBLEMS", below.

- Emanating from the end of the offset vector is a "baseline", whose direction is specified as an angle in degrees ("0.", "90.", "180.", or "270.", measured counter-clockwise from horizontal). The text lines of a label are written parallel to, and in the same direction as, the baseline.
• A centering option for each label determines whether the left edges, the
centers, or the right edges, of the text lines are aligned with the end
of the offset vector.

Each label may contain one or more text lines (or none). The total number of
text lines in all labels must not exceed n - normally, n = 16. Each of the
text lines is defined as follows:

• Each text line has an integral position number which distinguishes it
from every other line in the same label. Multiples of "100." are recom-
mended. Lines with positive position numbers are drawn above the label
baseline, lines with negative position numbers below the label baseline.
A line with position number "0." is centered on the label baseline.
"Above" and "below" are defined here from the viewpoint of a reader of
the label. The position numbers of the lines in a label specify the
order in which the lines appear - strictly decreasing from top to bottom
- but do not determine the interline spacing, which is set by AUTOGRAPH
itself.

• Each line has a "suppression flag", which may be set so as to enable or
disable drawing of the line.

• Each line has a character-width specifier, stated as a fraction of the
smaller dimension of the grid window.

• The text of each line is defined by a character string and a count of the
number of characters in the string - normally determined by AUTOGRAPH
itself.

Note: The (EDITOR-style) string replacements

`FLLB(10,8)` = `FLLB(10,m)`
`QBIM / 8.` = `QBIM / m.`

where m is greater than or equal to 5, may be applied to the AUTOGRAPH source
file to provide for a maximum of m labels. Similarly, the string replacements

`FLLN(6,16)` = `FLLN(6,n)`
`QNIM / 16.` = `QNIM / n.`

where n is greater than or equal to 5, may be applied to the AUTOGRAPH source
file to provide for a maximum of n lines.

2.31 THE PREDEFINED LABELS. The section "PARAMETERS" describes in detail
four "predefined" labels, named "R", "L", "B", and "T". Each of these labels
lies along one of the four edges of the grid window - the left edge, the right
edge, the bottom edge, or the top edge.

The predefined labels greatly simplify the task of generating labels along the
edges of the grid window. For example, if you want a "header label" above the
grid window, you need only specify the desired character string to define the
text of line number "100." of the label named "T".
The default definitions of the predefined labels specify a label reading "X" below the grid window and a label reading "Y" to the left of the grid window. See examples 5 and 7 and the final example, in the section "EXAMPLES".

2.32 THE PARAMETER GROUP 'LABEL.'. The parameter group 'LABEL.' contains $10m+3$ parameters - normally, $m=8$. Together with the parameters in the group 'LINE.', they define the informational labels to be drawn by a call to the routine AGBACK. The parameters in the group 'LABEL.' are as follows:

- The parameter 'LABEL/CONTROL.' may be given the value "-1." to delete all currently-defined labels, the value "0." to temporarily disable the drawing of labels, the value "1." to enable the drawing of labels and prevent the shrinkage of labels when overlap problems arise, or the value "2." to enable the drawing of labels and allow shrinkage. The default value is "2.". See examples 5 and 6, in the section "EXAMPLES".

- The parameter 'LABEL/BUFFER/LENGTH.' should not normally be set by a user program. Its value is $m$, the maximum number of labels AUTOGRAPH can handle.

- The subgroup 'LABEL/BUFFER/CONTENTS.' consists of $10m$ words, in which the label definitions are stored. Normally, a user program should not attempt to store values in this block directly. See the paragraph "HOW TO ACCESS A LABEL DEFINITION", below.

- The parameter 'LABEL/NAME.' is used in the process of accessing a label definition. It functions as a switch, pointing to the label definition currently being accessed.

See examples 5 and 7 and the final example, in the section "EXAMPLES".

See the section "PARAMETERS" for further information about these parameters.

2.33 THE PARAMETER GROUP 'LINE.'. The parameter group 'LINE.' contains $6n+4$ parameters - normally, $n=16$. They define the lines belonging to the various labels. The parameters in the group 'LINE.' are as follows:

- The parameters 'LINE/MAXIMUM.' and 'LINE/END.' define the assumed maximum line length (default - 40 characters) and the line end character (default - '$'). These parameters come into play when a user program defines the text of a line. The character string tendered by the user is assumed to be of maximum length; if it is really shorter than that, it must be followed by the line end character. See example 8 and the final example, in the section "EXAMPLES".

- The parameter 'LINE/BUFFER/LENGTH.' should not normally be set by a user program. Its value is $n$, the maximum number of lines AUTOGRAPH can handle.

- The subgroup 'LINE/BUFFER/CONTENTS.' consists of $6n$ words, in which the line definitions are stored. Normally, a user program should not attempt to store values in this block directly. See the paragraph "HOW TO ACCESS
A LINE DEFINITION", below.

- The parameter 'LINE/NUMBER.' is used in the process of accessing a line definition. It functions as a switch, pointing to the line definition currently being accessed.

See the section "PARAMETERS" for further information about these parameters.

See examples 5 and 7 and the final example, in the section "EXAMPLES".

2.34 ACCESSING A LABEL DEFINITION. To access a label definition, a user program must first execute an AGSETC call to store the name of the label as the value of 'LABEL/NAME.' Such a call does not actually store the name as the value of that parameter. Instead, it causes the label buffer to be searched for the definition of the named label. If that definition is not found, a default definition is made up and inserted in the label buffer. In any case, the index of the definition is floated and stored as the value of the parameter 'LABEL/NAME.'.

Once 'LABEL/NAME.' has been set in this manner, the parameter group name 'LABEL/DEFINITION.' and subgroup names of the form 'LABEL/DEFINITION/...' may be used to access the parameters defining the label. These parameters are as follows:

- The parameter 'LABEL/DEFINITION/SUPPRESSION.' may be given the value "-2." to delete the label and all of its lines, the value "-1." to delete the lines of the label but leave the label itself defined, the value "0." to enable drawing of the label, and the value "1." to temporarily suppress drawing of the label. It has the default value "0.". When a label is deleted, 'LABEL/NAME.' and 'LINE/NUMBER.' become undefined; similarly, when the lines of a label are deleted, 'LINE/NUMBER.' becomes undefined.

- The parameters 'LABEL/DEFINITION/BASEPOINT/X.' and '...Y.' specify the coordinates, in the grid coordinate system, of the label's basepoint. The default basepoint is (.5,.5).

- The parameters 'LABEL/DEFINITION/OFFSET/X.' and '...Y.' specify the components of the label's offset vector, as signed fractions of the smaller dimension of the grid window. The default vector has zero components.

- The parameter 'LABEL/DEFINITION/ANGLE.' specifies the angle ("0.", "90.", "180.", or "270.") at which the label's baseline emanates from the end of its offset vector. The default angle is "0.”.

- The parameter 'LABEL/DEFINITION/CENTER.' has the value "-1." to align the left ends, the value "0." to align the centers, and the value "+1." to align the right ends, of the lines of the label with the end of its offset vector. The default value is "0.".

- The parameters 'LABEL/DEFINITION/LINES.' and 'LABEL/DEFINITION/INDEX.' are not normally set by a user program; they are maintained by AUTOGRAPH. The former specifies the number of lines belonging to the label and the
latter specifies the index (in the line buffer) of the definition of the first line belonging to the label. A default label has no lines – both of these parameters are zeroed.

See the section "PARAMETERS" for further information about these parameters.

See examples 5 and 7 and the final example, in the section "EXAMPLES".

2.35 ACESSING A LINE DEFINITION. To access the definition of one of the lines of a label, a user program must first access the label definition by setting "LABEL/NAME.", as described above. Then, it must execute an AGSETP (or AGSETT) call to store the number of the desired line as the value of "LINE/NUMBER.". Such a call does not actually store the specified number as the value of that parameter. Instead, it causes the line buffer to be searched for the definition of the desired line. If that definition is not found, a default definition is made up, inserted in the line buffer, and added to the linked list of definitions of lines belonging to the label. In any case, the index of the definition is floated and stored as the value of "LINE/NUMBER."

Once "LINE/NUMBER." has been set in this manner, the parameter group name "LINE/DEFINITION." and subgroup names of the form "LINE/DEFINITION/..." may be used to access the parameters defining the line. These parameters are as follows:

• The parameter "LINE/DEFINITION/SUPPRESSION." may be given the value "-1." to delete the line, the value "0." to enable drawing of the line, and the value "+1." to temporarily disable drawing of the line. It has the default value "0.". When a line is deleted, "LINE/NUMBER." becomes undefined.

• The parameter "LINE/DEFINITION/CHARACTER-WIDTH." specifies the desired width of each character in the line, stated as a fraction of the smaller dimension of the grid window. The default width is ".015".

• The parameter "LINE/DEFINITION/TEXT." identifies the character string comprising the text of the label. The default value is a single blank.

• The parameter "LINE/DEFINITION/LENGTH." specifies the length of the character string. The default value is a "1.".

• The parameter "LINE/DEFINITION/INDEX." is not normally set by a user program. It is maintained by AUTOGRAPH and specifies the index (in the line buffer) of the next line belonging to the label.

See the section "PARAMETERS" for further information about these parameters. See examples 5 and 7 and the final example, in the section "EXAMPLES".

Note: As a convenience to the user, an AGSETC call to set "LINE/DEFINITION/TEXT." sets both the parameters defining the text of the label – "...TEXT." and "...LENGTH." – the latter being computed by examining the string. The string must be no longer than the length specified by the value of "LINE/MAXIMUM." and, if shorter, it must be followed by the
2.36 **THE LABEL BOXES.** Each informational label is considered to lie in one of six "label boxes", as follows:

- **Box 1** lies to the left of the grid window. It contains all labels which have a basepoint on the left edge of the grid window and a leftward-pointing offset vector.

- **Box 2** lies to the right of the grid window. It contains all labels which have a basepoint on the right edge of the grid window and a rightward-pointing offset vector.

- **Box 3** lies below the grid window. It contains all labels which have a basepoint on the bottom edge of the grid window and a downward-pointing offset vector.

- **Box 4** lies above the grid window. It contains all labels which have a basepoint on the top edge of the grid window and an upward-pointing offset vector.

- **Box 5** lies in the interior of the grid window. It contains all labels which have a basepoint on some edge of the grid window and an inward-pointing offset vector.

- **Box 6** covers the entire grid window and contains all of the remaining labels.

Three restrictions must be observed by the user: First, no label's basepoint may have coordinates (0.,0.), (0.,1.), (1.,0.), or (1.,1.); these corner points must be avoided. Second, no portion of any label in boxes 1 through 4 may lie inside the grid window. Third, no portion of any label in box 5 may lie outside the grid window.

The label-box concept is important in handling overlap problems, which are discussed in the next paragraph.

2.37 **BACKGROUND OVERLAP PROBLEMS.** The responsibility for avoiding background overlap problems might reasonably have been placed squarely on the shoulders of the user, except for one unpleasant fact: numeric labels are unpredictable critters. Accordingly, AUTOGRAPH accepts a part of the burden.

In attempting to keep the numeric labels on a given axis from overlapping each other, AUTOGRAPH may shrink and/or reorient them. Either or both of these actions may be suppressed by the user by resetting "AXIS/s/CONTROL.". If a problem still exists, some of the labels may be omitted — perhaps leaving only every second one, every third one, every fourth one, etc.

Informational labels are positioned by the user along the edges of the grid window as if numeric labels did not exist. AUTOGRAPH takes the following actions in attempting to prevent the informational labels from overlapping the numeric labels on any axis:
1. Box 1 labels (to the left of the grid window) are moved leftward, box 2 labels (to the right of the grid window) are moved rightward, box 3 labels (below the grid window) are moved downward, box 4 labels (above the grid window) are moved upward, and box 5 labels (inside the grid window) are moved inward. Box 6 labels are not moved.

2. If, during step 1, a label is shoved outside the graph window by the numeric labels on some axis, those numeric labels may be shrunk and/or re-oriented, as allowed by the user's setting of 'AXIS/s/CONTROL.'.

3. If one or more of the labels in a given box still lies partly outside the graph window, the labels in that box may be shrunk, depending on the current setting of 'LABEL/CONTROL.' Each label in the box shrinks toward the end of its offset vector.

4. If one or more of the labels in boxes 1 through 4 still lies partly outside the graph window, all of the labels in that box may be moved inward, shoving numeric labels ahead of them - onto, and perhaps across, an axis.

The algorithms used to do all of this are not perfect; if pushed too severely, they may fail to produce an esthetically pleasing or even minimally acceptable graph. In such cases, the user must take remedial action.

Note: None of the actions described above modify any of the parameters except 'AXIS/s/NUMERIC/ANGLE/LST.', which may be negated by subtracting a multiple of "360.". Also, no label is shrunk to less than a readable size.

2.38 DASHED-LINE PATTERNS FOR CURVES. The subroutine AGCURV draws curves (one per call). It does this by issuing calls to the routines DASHD, FRSTD, VECTD, and LASTD, in the DASHCHAR package. Each curve may thus be drawn using its own particular dashed-line pattern. One of AGCURV's arguments, called KDSH, specifies the dashed-line pattern to be used for a given curve:

- If KDSH is zero, the caller is assumed to have done his own call to DASHD. AGCURV does not call it.

- If KDSH is non-zero, AGCURV calls DASHD.

- If KDSH is positive, its value (modulo n) specifies one of n "user" patterns, defined by the parameter group named 'DASH.'. See example 7, in the section "EXAMPLES".

- If KDSH is negative, its absolute value (modulo 26) specifies one of 26 "alphabetic" patterns. The curve is drawn using a solid line which is interrupted periodically by the selected letter of the alphabet. See example 8, in the section "EXAMPLES".

The nature of the "user" set of dashed-line patterns is discussed in the next paragraph.

2.39 THE PARAMETER GROUP 'DASH.'. The following parameters all belong to the group 'DASH.':
• The parameter 'DASH/SECTOR.' specifies the type of dashed-line patterns to be used by the routines EZMY and EZMXY (EZY and EZXY always use the first of the "user" patterns). If the value of 'DASH/SECTOR.' is zero or negative, the alphabetic set of (26) dashed-line patterns will be used; if its value is positive and has magnitude "n", then the first "n" of the "user" patterns will be used. The default value is "1."

• The parameter 'DASH/LENGTH.' specifies the assumed length of character-string dashed-line patterns; it must be set to the proper value prior to any AGSETC call setting one of the "user" dashed-line patterns 'DASH/PATTERN/n.'. The default value is "8."

• The parameter 'DASH/CHARACTER.' specifies the curve length devoted to a character other than a dollar sign or a quote in a character-string dashed-line pattern, stated as a fraction of the smaller side of the grid window. The default value is ".010".

• The parameter 'DASH/DOLLAR-QUOTE.' specifies the curve length devoted to a dollar sign (a solid section of the line) or a quote (a gap in the line), stated as a fraction of the smaller side of the grid window. The default value is ".010".

• Each of the parameters 'DASH/PATTERN/1.', 'DASH/PATTERN/2.', etc., up to 'DASH/PATTERN/26.' defines one of the dash patterns in the "user" group. Either of the routines AGSETT or AGSETF may be used to give one of these parameters a positive integral value between 0. and 65535., inclusive, in which case the low-order 16 bits of it are interpreted as a dash pattern; the 0 and 1 bits represent "pen-up" and "pen-down" segments three plotter units long. The routine AGSETC may be used to (in effect) store a character string (of the length specified by 'DASH/LENGTH.' as the value of one of these parameters. In such a character string, a quote represents a "pen-up" segment, a dollar sign represents a "pen-down" segment, and every other character is simply drawn as a part of the line. The default value of all 26 parameters is "65535.".

See examples 7 and 8, in the section "EXAMPLES".

2.40 PATTERNS USED BY EZY, EZXY, EZMY, AND EZMXY. Each of the routines EZY and EZXY, which draw one curve per call, calls AGCURV with KDSH equal to 1, specifying the use of the first of the "user-defined" set of dashed-line patterns (default - a solid line) for the single curve to be drawn.

Each of the routines EZMY and EZMXY, which draw one or more curves per call, calls AGCURV with KDSH equal to ISIGN(I,IDSH), where I is the number of the curve being drawn and ISH is the integral value of "DASH/SECTOR.". This parameter has the default value "1.", specifying the use of the "user" set of dashed-line patterns (default - solid lines); the value "-1." specifies the use of the "alphabetic" set.

2.41 WINDOWING OF CURVES. The parameter 'WINDOW.' has the default value "0.". If it is set to a "1." by a user program, curves subsequently drawn by the routine AGCURV are "windowed". This means that only those portions lying inside the grid window are drawn; the effect is as if one were viewing the
curve through an actual window.
See example 7, in the section "EXAMPLES".

2.42 USE OF PWRITX BY AUTOGRAPH. Normally, the routine PWRIT is used for all characters drawn by AUTOGRAPH. Actually, a routine AGPWRT is called; the default version of that routine just passes its arguments on to PWRIT. Trying to use PWRITX instead poses some problems. Because "function codes" may be used in a text string passed to PWRITX, the length of the string cannot be taken to match the actual number of characters to be drawn; moreover, PWRITX does not use the same plotter width for each character in a string. Thus, AUTOGRAPH cannot properly predict where on a graph a label drawn by PWRITX lies, which interferes with its handling of overlap problems; also, strings which are positioned relative to an end-point may not be properly aligned.

Nevertheless, there is a way to use PWRITX. The XLIB file AGUPWRTX contains a version of AGPWRT which does it. A string which is centered relative to a given position (like the "x-axis label", the "y-axis label", or the "graph label") is drawn directly, in its entirety, by PWRITX, and may therefore contain function codes to get Greek characters, subscripts, superscripts, etc. A string which is positioned relative to one end (which includes all numeric labels) is drawn by passing one character at a time to PWRITX, so that the same plotter width will be used for each; function codes must not be included in such strings. The results, while not as pleasing as one would normally expect from PWRITX, are more than just acceptable.

On the Cray, use the following JCL to get the source for this version of AGPWRT and compile it, so that it will replace the default version:

```
GETSRC,LIB=XLIB,FILE=AGUPWRTX,L=UPWRTX.
CFT,I=UPWRTX,L=0.
```

In my opinion, the "duplex" character set of PWRITX is far superior to the "complex" set. At present, one uses it by incorporating the following code at the beginning of one’s program, prior to any call to PWRITX (directly or indirectly):

```
COMMON /USER/ MODE
:::
MODE = 1
```

2.43 VARYING INTENSITIES, COLORS, ETC.. Three routines - AGCHAX, AGCHCU, and AGCHIL (the default versions of which do nothing) - are provided solely to be replaced by the user; the replacement versions may change intensities, line widths, colors, line styles, etc., for selected portions of a graph. Each is called just before an object is to be drawn and again just after it has been drawn, with arguments enabling the user version to completely identify what the current situation is and to make the appropriate calls. AGCHAX handles objects which are parts of axes, AGCHCU handles curves, and AGCHIL handles informational labels.

See the descriptions of these routines, in the section "ROUTINES".
2.44 NON-STANDARD NUMERIC LABELS. The routine AGCHNL (the default version of which does nothing) is called just after the character form of a numeric label has been constructed and just before it is to be drawn. The user may supply a version of this routine to transform selected numeric labels in any desired fashion and return them to AUTOGRAPH. This feature may be used to label an axis with the names of the months, Roman numerals, etc.

See the description of the routine AGCHNL, in the section "ROUTINES" and example 10, in the section "EXAMPLES".

2.45 SCATTERGRAMS AND HISTOGRAMS. Scattergrams, histograms, and other such specialized "graphs" are not directly provided for by AUTOGRAPH. Standard procedure is to suppress the advancing of the frame and the drawing of curves by EZ..., call EZ... with the appropriate x and y data to generate the background, draw the desired objects on that background, and then advance the frame.

See examples 11 and 12, in the section "EXAMPLES".

2.46 GKS CONSIDERATIONS. From the user's point of view, the GKS version of AUTOGRAPH is identical to the NSPP (NCAR System Plot Package) version; in fact, they are very nearly identical from the implementor's point of view. The GKS version depends on the SPPS (System Plot Package Simulator); as of this writing, it has not been determined whether or not the use of SPPS will require special JCL in the user's program. Check with a consultant to determine the current situation.

The examples given in the section "EXAMPLES" must be modified slightly to run with the GKS version of AUTOGRAPH. Each program must start by executing a call to OPNGKS and end by executing a call to CLSGKS. In example 11, the fourth argument in the call to POINTS must be modified; I recommend changing it from a '+' to a -2.
3. ROUTINES

This section describes all of the AUTOGRAPH routines of interest to the user. With two exceptions, they are subroutines rather than functions.

The subroutines EZY, EZXY, EZMY, and ERMXY provide a quick-and脏 diets graph-drawing capability. The appearance of a graph drawn by one of these routines may be changed drastically by changing the values of primary control parameters.

The subroutines ANOTAT and DISPLA are provided principally for historical reasons. Each allows the user to provide new values for certain primary control parameters. These parameters may also be set by calls to lower-level routines.

The subroutines AGSETP, AGSETF, AGSETI, and AGSETC allow the user to set the values of parameters.

The subroutines AGGETP, AGGETF, AGGETI, and AGGETC allow the user to get the current values of parameters.

The subroutine AGSTUP must be called prior to calling AGBACK or AGCURV, both initially and after making any change in the primary control parameters. It examines the primary control parameters for consistency and computes the values of required secondary control parameters.

The subroutine AGBACK is called to draw a background.

The subroutine AGCURV is called to draw a single curve.

The subroutines AGSAVE and AGRSTR are called to save/restore the current state (commonly the default state) of AUTOGRAPH.

The function AGBNCH may be used to obtain the 16-character dash pattern which is equivalent to a specified 16-bit integer dash pattern.

The function AGDSHN may be used to obtain the 16-character name of a particular specified dash-pattern parameter.

The subroutine AGUJL is called by AUTOGRAPH to do the mapping from the user system to the label system (or vice-versa) along the four axes. The default version may be replaced by the user to obtain a desired mapping.

The subroutines AGCHAX, AGCHCU, and AGCHIL are called by AUTOGRAPH just before and just after drawing a particular element of a graph. The default do-nothing versions may be replaced by a user in order to obtain desired effects (color, different line styles, etc.).

The subroutine AGCHNL is called by AUTOGRAPH just after the character representation of a numeric label has been generated and just before it is to be drawn. The default do-nothing version may be replaced by a user in order to change the numeric labels in a desired way (to get names of months, Roman numerals, etc.).
The subroutine AGPWdR is called by AUTOGRAPH to draw a character string. The default version just calls the plot-package routine PWRIT. This routine may be replaced by a user version which calls PWRITX, PWRITY, or some other character-drawer.

Note: User versions of AGUTOL, AGCHAX, AGCHCU, AGCHIL, AGCHNL, and AGPWRT should not call any other AUTOGRAPH routine. No such call will have a useful effect, and, at worst, an infinite loop may result.

3.1 EZY(YDRA,NPTS,GLAB). Draws, in a manner determined by the current values of the control parameters, a complete graph of a single curve through the points (I,YDRA(I)), for I from 1 to NPTS. The argument GLAB may be used to specify a "graph label", to be placed at the top of the graph.

3.1.1 EZY - USAGE. If the default values of the parameters are unchanged, calling EZY produces a graph having the following appearance: A "perimeter" background outlines a grid window which is 8/10 the width and 8/10 the height of the plotter frame and positioned slightly above and to the right of center within it. Each edge of the perimeter has short inward-pointing major and minor tick marks, with major tick marks occurring at the ends of each edge. Numeric labels below major tick marks on the bottom edge of the perimeter, increasing in value from left to right, show the linear mapping of values of I onto the horizontal (x) axis of the graph; below them is the label "X". Numeric labels to the left of major tick marks on the left edge of the perimeter, increasing in value from bottom to top, show the linear mapping of values of YDRA(I) onto the vertical (y) axis of the graph; to the left of them is the label "Y". Above the perimeter is the label specified by "GLAB", if any. The curve itself is drawn as a solid line within the perimeter. A frame advance is done after the graph is drawn.

See example 1, in the section "EXAMPLES".

The appearance of a graph drawn by EZY may change greatly in response to parameter changes. See the routines ANOTAT, DISPLA, and AGSETP, below, and the section "PARAMETERS".

3.1.2 EZY - ARGUMENTS. REAL YDRA(NPTS); INTEGER NPTS; CHARACTER*(*) GLAB.

YDRA is a one-dimensional array of NPTS floating-point numbers, each of which defines the user coordinates of a point (FLOAT(I),YDRA(I)) on the desired curve. The current value of 'NULL/1.' (default value "1.E36") may be used in YDRA to signal missing points; curve segments on either side of a missing point are omitted.

NPTS is the number of curve points defined by the array YDRA.

GLAB is a character expression defining a new "graph label". (If the first character of this expression is "CHAR(0)", no new "graph label" is defined; the current one will continue to be used.) A character string defining a new graph label must either be of the exact length specified by the current value of 'LINE/MAXIMUM.' (default - 40 characters), or shorter; if shorter, it must
be terminated by the character defined by the current value of 'LINE/END.'
(default - a ' $'). The string becomes the new text of line number 100 of the
label named "T".

\[ \text{3.2 } \text{EZXY(XDRA, YDRA, NPTS, GLAB).} \] Draws, in a manner determined by the current
values of the control parameters, a complete graph of a single curve through
the points (XDRA(I), YDRA(I)), for I from 1 to NPTS. The argument GLAB may be
used to specify a "graph label", to be placed at the top of the graph.

\[ \text{3.2.1 } \text{EZXY - USAGE.} \] If the default values of the parameters are unchanged,
calling EZXY produces a graph having the following appearance: A "perimeter"
background outlines a grid window which is 8/10 the width and 8/10 the height
of the plotter frame and positioned slightly above and to the right of center
within it. Each edge of the perimeter has short inward-pointing major and
minor tick marks, with major tick marks occurring at the ends of each edge.
Numeric labels below major tick marks on the bottom edge of the perimeter,
increasing in value from left to right, show the linear mapping of values of
XDRA(I) onto the horizontal (x) axis of the graph; below them is the label
"x". Numeric labels to the left of major tick marks on the left edge of the
perimeter, increasing in value from bottom to top, show the linear mapping of
values of YDRA(I) onto the vertical (y) axis of the graph; to the left of them
is the label "y". Above the perimeter is the label specified by "GLAB", if
any. The curve itself is drawn as a solid line within the perimeter. A frame
advance is done after the graph is drawn.

See example 2, in the section "EXAMPLES".

The appearance of a graph drawn by EZXY may change greatly in response to
parameter changes. See the routines ANOTAT, DISPLA, and AGSETP, below, and
the section "PARAMETERS".

\[ \text{3.2.2 } \text{EZXY - ARGUMENTS.} \] REAL XDRA(NPTS), YDRA(NPTS); INTEGER NPTS; CHARACTER*(*) GLAB.

XDRA is a one-dimensional array of NPTS floating-point numbers, defining the x
coordinates of points on the curve.

YDRA is a one-dimensional array of NPTS floating-point numbers, defining the y
coordinates of points on the curve.

The points on the curve have coordinates (XDRA(I), YDRA(I)), for I from 1 to
NPTS. The current value of 'NULL/1.' (default value "1.E36") may be used to
signal missing data in these arrays. If either coordinate of a point is miss-
ing, the point is considered to be missing; curve segments on either side of a
missing point are not drawn. Note: Because all non-missing coordinates are
used in figuring the minimum and maximum user values along a given axis, it is
safest to mark both coordinates as "missing".

NPTS is the number of curve points defined by the arrays XDRA and YDRA.
GIAB is a character expression defining a new "graph label". (If the first character of this expression is "CHAR(0)", no new "graph label" is defined; the current one will continue to be used.) A character string defining a new graph label must either be of the exact length specified by the current value of "LINE/MAXIMUM." (default - 40 characters), or shorter; if shorter, it must be terminated by the character defined by the current value of "LINE/END." (default - a "$"). The string becomes the new text of line number 100 of the label named "T".

3.3 EZMY(YDRA, IDXY, MANY, NPTS, GLAB). Draws, in a manner determined by the current values of the control parameters, a complete graph of one or more curves, each defined by a set of points (I,YDRA(I,J)) (or (I,YDRA(J,I)), depending on the current value of "ROW."), for I from 1 to NPTS. The curve number J runs from 1 to MANY. The argument GLAB may be used to specify a "graph label", to be placed at the top of the graph.

3.3.1 EZMY - USAGE. If the default values of the parameters are unchanged, calling EZMY produces a graph having the following appearance: A "perimeter" background outlines a grid window which is 8/10 the width and 8/10 the height of the plotter frame and positioned slightly above and to the right of center within it. Each edge of the perimeter has short inward-pointing major and minor tick marks, with major tick marks occurring at the ends of each edge. Numeric labels below major tick marks on the bottom edge of the perimeter, increasing in value from left to right, show the linear mapping of values of I onto the horizontal (x) axis of the graph; below them is the label "X". Numeric labels to the left of major tick marks on the left edge of the perimeter, increasing in value from bottom to top, show the linear mapping of values of YDRA(I,J) onto the vertical (y) axis of the graph; to the left of them is the label "Y". Above the perimeter is the label specified by "GLAB", if any. The curves themselves are drawn as solid lines within the perimeter. A frame advance is done after the graph is drawn.

See example 3, in the section "EXAMPLES".

The appearance of a graph drawn by EZMY may change greatly in response to parameter changes. See the routines ANOTAT, DISPLA, and AGSETP, below, and the section "PARAMETERS".

3.3.2 EZMY - ARGUMENTS. REAL YDRA(IDXY,MANY) or YDRA(IDXY,NPTS) (depending on the current value of "ROW."); INTEGER IDXY, MANY, NPTS; CHARACTER(*) GLAB.

YDRA is a two-dimensional array of curve-point y coordinates. The current value of "NULL/1." (default value "1.E36") may be used in YDRA to signal missing points; curve segments on either side of a missing point are omitted.

If "ROW." is positive (the default), the first subscript of YDRA is a point number and the second is a curve number. If "ROW." is negative, the order of the subscripts is reversed ("row-wise", as opposed to "column-wise", storage).

IDXY is the first dimension of the array YDRA, required by EZMY in order to index the array properly.
If 'ROW.' is positive (the default), IDXY must be greater than or equal to NPTS; otherwise, IDXY must be greater than or equal to MANY.

**MANY** is the number of curves to be drawn by EZMY.

**NPTS** is the number of points defining each curve to be drawn by EZMY.

**GLAB** is a character expression defining a new "graph label". (If the first character of this expression is "CHAR(0)", no new "graph label" is defined; the current one will continue to be used.) A character string defining a new graph label must either be of the exact length specified by the current value of 'LINE/MAXIMUM.' (default - 40 characters), or shorter; if shorter, it must be terminated by the character defined by the current value of 'LINE/END.' (default - a "$"). The string becomes the new text of line number 100 of the label named 'T'.

---

### 3.4 EZMXY (XDRA, YDRA, IDXY, MANY, NPTS, GLAB)

Draws, in a manner determined by the current values of the control parameters, a complete graph of one or more curves, each defined by a set of points (XDRA(I),YDRA(I,J)) or (XDRA(J),YDRA(I,J)) or (XDRA(I,J),YDRA(J,I)), depending on the current value of 'ROW.'), for I from 1 to NPTS. The curve number J runs from 1 to MANY. The argument GLAB may be used to specify a "graph label", to be placed at the top of the graph.

#### 3.4.1 EZMXY - USAGE

If the default values of the parameters are unchanged, calling EZMXY produces a graph having the following appearance: A perimeter background outlines a grid window which is 8/10 the width and 8/10 the height of the plotter frame and positioned slightly above and to the right of center within it. Each edge of the perimeter has short inward-pointing major and minor tick marks, with major tick marks occurring at the ends of each edge. Numeric labels below major tick marks on the bottom edge of the perimeter, increasing in value from left to right, show the linear mapping of values of XDRA(I) onto the horizontal (x) axis of the graph; below them is the label "X". Numeric labels to the left of major tick marks on the left edge of the perimeter, increasing in value from bottom to top, show the linear mapping of values of YDRA(I,J) onto the vertical (y) axis of the graph; to the left of them is the label "Y". Above the perimeter is the label specified by "GLAB", if any. The curves themselves are drawn as solid lines within the perimeter. A frame advance is done after the graph is drawn.

See example 4, in the section "EXAMPLES".

The appearance of a graph drawn by EZMXY may change greatly in response to parameter changes. See the routines ANOTAT, DISPLA, and AGSETP, below, and the section "PARAMETERS".

#### 3.4.2 EZMXY - ARGUMENTS

REAL XDRA(NPTS) or XDRA(IDXY,MANY) or XDRA(IDXY,NPTS), depending on the current value of 'ROW.'; REAL YDRA(IDXY,MANY) or YDRA(IDXY,NPTS), depending on the current value of 'ROW.'; INTEGER IDXY, MANY, NPTS; CHARACTER*(*) GLAB.
XDRA is a one-dimensional or two-dimensional array of curve-point x coordinates. The current value of 'NULL/1.' (default value "1.E36") may be used in XDRA to signal missing points; curve segments on either side of a missing point are not drawn. Note: Because all non-missing coordinates are used in figuring the minimum and maximum user values along a given axis, it is safest to mark both coordinates as "missing".

If "ROW." has the absolute value "1." (the default), XDRA is singly-dimensioned. It is subscripted by point number.

If "ROW." has the absolute value "2." or greater, XDRA is doubly-dimensioned. It is subscripted by point number and curve number, in that order if "ROW." is positive (the default), in the reverse order if "ROW." is negative.

YDRA is a two-dimensional array of curve-point y coordinates. The current value of 'NULL/1.' (default value "1.E36") may be used in YDRA to signal missing points; curve segments on either side of a missing point are not drawn. Note: Because all non-missing coordinates are used in figuring the minimum and maximum user values along a given axis, it is safest to mark both coordinates as "missing".

If "ROW." is positive (the default), YDRA is subscripted by point number and curve number, in that order; otherwise, YDRA is subscripted by curve number and point number, in that order.

IDXI is the first dimension of the arrays XDRA (if it is doubly-dimensioned) and YDRA (unconditionally), required by EZMXY in order to index these arrays properly.

If "ROW." is positive (the default), IDXY must be greater than or equal to NPTS; otherwise, IDXY must be greater than or equal to MANY.

MANY is the number of curves to be drawn by EZMXY.

NPTS is the number of points defining each curve to be drawn by EZMXY.

GLAB is a character expression defining a new "graph label". (If the first character of this expression is "CHAR(O)", no new "graph label" is defined; the current one will continue to be used.) A character string defining a new graph label must either be of the exact length specified by the current value of "LINE/MAXIMUM." (default - 40 characters), or shorter; if shorter, it must be terminated by the character defined by the current value of "LINE/END." (default - a "$\)$. The string becomes the new text of line number 100 of the label named "T".

3.5 ANOTAT(XLAB,YLAB,BACSET,,NDSH,D6). Changes the values of certain primary control parameters, purportedly having to do with "annotation" of a graph.
3.5.1 **ANOTAT** - **USAGE.** The subroutine **ANOTAT** is provided principally for historical reasons. Each of the parameters referenced by its argument list can be set individually by means of the routines **AGSETP**, **AGSETF**, and/or **AGSETI**. In fact, **ANOTAT** is implemented using calls to these routines.

See example 8, in the section "EXAMPLES".

3.5.2 **ANOTAT** - **ARGUMENTS.** INTEGER LBAC, LSET, NDSH; CHARACTER(*) XLAB, YLAB, DSHC(NDSH).

**XLAB** is a character expression defining a new "x-axis label". (If the first character of this expression is "CHAR(0)", no new "x-axis label" is defined; the current one will continue to be used.) A character string defining a new x-axis label must either be of the exact length specified by the current value of "LINE/MAXIMUM." (default - 40 characters), or shorter; if shorter, it must be terminated by the character defined by the current value of "LINE/END." (default - a '$'). The string becomes the new text of line number -100 of the label "B".

**YLAB** is a character expression defining a new "y-axis label". (If the first character of this expression is "CHAR(0)", no new "y-axis label" is defined; the current one will continue to be used.) A character string defining a new y-axis label must either be of the exact length specified by the current value of "LINE/MAXIMUM." (default - 40 characters), or shorter; if shorter, it must be terminated by the character defined by the current value of "LINE/END." (default - a '$'). The string becomes the new text of line number 100 of the label "L".

**LBAC**, if non-zero, has the integer value 1, 2, 3, or 4, the floating-point equivalent of which is to become the new value of "BACKGROUND.". (If LBAC is zero, no change is to be made in the current value.) The value "1" specifies a perimeter background, the value "2" a grid background, the value "3" a half-axis background, and the value "4" no background at all.

See the discussion of "BACKGROUND.", in the section "PARAMETERS".

**LSET**, if non-zero, is an integer having the absolute value 1, 2, 3, or 4, the floating-point equivalent of which is to be stored (by means of a call to **AGSETI**) as the new value of "SET.". If LSET is zero, no change is to be made in the current value of "SET.".

See the discussion of "SET.", in the section "PARAMETERS".

**NDSH**, if zero, specifies that no change is to be made in the parameters which specify the dashed-line patterns to be used for curves. If NDSH is non-zero, it specifies an integer value whose floating-point equivalent is to be stored as the new value of "DASH/SELECTOR." (which has the default value "1.0").

If NDSH is negative, "DASH/SELECTOR." is set negative, forcing **EZMY** and **EZMXY** to use internally-defined "alphabetic" patterns for the **MANY** curves drawn ("A" for the first, "B" for the second, ..., "Z" for the 26th, "A" for the 27th, etc.). The routines **EZY** and **EZXY** are unaffected.
If NDSH is greater than zero, it must be less than or equal to 26, and the next argument, DSHC, must contain NDSH dashed-line patterns comprising the new "user" set of patterns. The fact that 'DASH/SELECTOR.' is set positive forces EZMY and EZMXY to use this set of patterns. (The routines EZY and EZXY always use the first pattern in this set.) The contents of the array DSHC are copied to storage local to AUTOGRAPH and pointers to them are installed as the values of 'DASH/PATTERNS/1.', '.../2.', etc.

See the discussion of 'DASH.', in the section "PARAMETERS".

DSHC is meaningful only when NDSH is greater than zero. In this case, it must be an array of NDSH character strings, each of the length specified by the current value of 'DASH/LENGTH.' Each character string represents a dashed-line pattern; dollar signs mean "pen down", quotes mean "pen up", and other characters mean "draw me".

See the discussion of 'DASH.', in the section "PARAMETERS".

3.6 DISPLA(LFRA,LROW,LTYP). Changes the values of certain primary control parameters, purportedly having to do with the "display" of a graph.

3.6.1 DISPLA - USAGE. The subroutine DISPLA is provided principally for historical reasons. Each of the parameters referenced by its argument list can be set individually by means of the routines AGSETP and/or AGSETI. In fact, DISPLA is implemented using calls to these routines.

See the final example, in the section "EXAMPLES".

3.6.2 DISPLA - ARGUMENTS. INTEGER LFRA, LROW, LTYP.

LFRA, if non-zero, has an integer value, the floating-point equivalent of which is to become the new value of 'FRAME.' If LFRA is zero, no change is to be made in the current value of 'FRAME.'.

The default value of 'FRAME.' is "1.", specifying that each of the routines EZY, EZXY, EZMY, and EZMXY is to do a frame advance after drawing a graph. The value "2." specifies that these routines should do no frame advance, the value "3." that they should do a frame advance before drawing a graph.

LROW, if non-zero, has an integer value, the floating-point equivalent of which is to become the new value of 'ROW.'. If LROW is zero, no change is to be made in the current value of 'ROW.'.

This parameter affects the way in which the routines EZMY and EZMXY interpret the arguments XDRA and YDRA, as follows:

If 'ROW.' is positive, the first subscript of YDRA is a point number and the second subscript is a curve number. If 'ROW.' is negative, the order of the subscripts is reversed (row-wise, rather than column-wise, storage).
If the absolute value of °ROW.° is "1.°, XDRA is singly-subscripted; its subscript is a point number. If the absolute value of °ROW.° is "2." or greater, XDRA is doubly-subscripted; the order of the subscripts is the same as for YDRA.

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

LTYP, if non-zero, is an integer specifying new values for °X/LOGARITHMIC.° and °Y/LOGARITHMIC.°. If LTYP is zero, no change is to be made in the current values.

The parameter °X/LOGARITHMIC.° has the default value "0.°, specifying a linear mapping of user x coordinates onto the horizontal axis of the grid window; it may be given either of the two values "-1." or "+1." to specify a logarithmic mapping. The value "-1." protects it from being reset as a side effect of setting °SET.". DISPLA generates the value "0." or "-1.°.

The parameter °Y/LOGARITHMIC.° is defined similarly and affects the mapping of user y coordinates onto the vertical axis of the grid window.

A non-zero LTYP resets these values, as follows:

<table>
<thead>
<tr>
<th>LTYP</th>
<th>°X/LOGARITHMIC.°</th>
<th>°Y/LOGARITHMIC.°</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td>2</td>
<td>linear</td>
<td>logarithmic</td>
</tr>
<tr>
<td>3</td>
<td>logarithmic</td>
<td>linear</td>
</tr>
<tr>
<td>4</td>
<td>logarithmic</td>
<td>logarithmic</td>
</tr>
</tbody>
</table>

3.7 AGSETP (TPGN, PURA, LURA). Allows a user program to reset the values of a group of parameters containing one or more elements.

3.7.1 AGSETP - USAGE. The subroutine AGSETP is called with a character string identifying a group of parameters (possibly a single parameter), an array containing new values of those parameters, and the length of the array, as arguments. It transfers the new values into the appropriate locations in the labeled common block AGCONP, thus modifying the effect of subsequent calls to other AUTOGRAPH routines.

If a single parameter is being set, one of the routines AGSETF, AGSETI, or AGSETC (which see, below) may be used instead.

When certain parameters are set individually, AGSETP takes further "special" action. For example, if °BACKGROUND.° is set, thereby requesting one of the standard types of backgrounds, AGSETP changes a number of other parameters in order to achieve the desired effect. The parameters which imply such special action are as follows:
SET.
BACKGROUND.
"NULL/1." and "NULL/2."
LABEL/CONTROL.
LABEL/NAME.
LABEL/DEFINITION/SUPPRESSION.
LINE/NUMBER.
LINE/DEFINITION/SUPPRESSION.
LINE/DEFINITION/TEXT.

See the section "PARAMETERS" for descriptions of the parameters whose values may be set.

3.7.2 AGSETP - ARGUMENTS. CHARACTER(*) TPGN; REAL FURA(LURA); INTEGER LURA.

TPGN is a character string of the form 'k(1)/k(2)/...k(n).', where each of the k(i)'s is a keyword. The keyword k(1) specifies a group of parameters, k(2) a subgroup of that group, k(3) a subgroup of that subgroup, etc. The whole string is the name of some group of parameters the user wishes to set.

For example, 'AXIS.' is the name of a 92-word group of parameters describing the four axes, 'AXIS/RIGHT.' is the name of a 23-word subgroup describing the right y axis,

'AXIS/RIGHT/INTERSECTION.'

is the name of a 2-word further subgroup describing the intersection of the right y axis with the bottom of the grid window, and

'AXIS/RIGHT/INTERSECTION/USER.'

is the name of a single parameter specifying the point of intersection of the right y axis with the bottom of the grid window as an x coordinate in the user coordinate system.

Obviously, these names can sometimes become rather long. There are various ways in which they may be shortened. First, since the fifth and following characters of each keyword are ignored, they may be omitted; this would shorten

'AXIS/RIGHT/INTERSECTION/USER.'

to

'AXIS/RIGH/INTE/USER.'

Even fewer characters may be used, as long as no ambiguity of interpretation arises. To be completely safe, use at least the first three characters of the group keyword and at least the first two characters of each subgroup keyword; this would shorten the example above to 'AXI/RI/IN/US.'. Moreover, certain group and subgroup keywords may be omitted entirely; for example, 'AXI/RI/IN/US.' may be shortened to 'RI/IN/US.'.
Names may also be lengthened in various ways in order to improve their readability. Blanks may be used as desired on either side of a keyword. Any sequence of characters not including a slash or a period may be inserted after a keyword, separated from it by at least one blank. For example, the name

`'DASH PATTERN / CHARACTER WIDTH'`

is equivalent to, and considerably more meaningful than,

`'DAS/CH.'` (or even `'DASH/CHARACTER.'`)

A complete list of the parameters may be found in the section "PARAMETERS", below.

**FURA** is a user array (of length LURA) containing the new values of the parameters in the group specified by TPGN, in the same order as they appear in the group.

All parameters have floating-point values (because of a portability problem which arose in implementing AGSETF). Those which represent intrinsically integral quantities have a value of the form "FLOAT(n)", where "n" is the integral quantity being represented. Some parameters intrinsically take on character-string values; the floating-point quantity stored as the value of such a parameter is typically an identifier allowing for later retrieval of the character string from a character storage area inside AUTOGRAPH. The routines AGSETC and AGGETC may be used to set/get the character-string values of such parameters.

**LURA** is the length of FURA (the number of floating-point elements in it). Its value may be less than, equal to, or greater than, the length of the parameter group specified by TPGN. The number of values transferred from FURA is the minimum of the two (but not less than one). This means that if, for example, you only wish to set the first two parameters of a 100-parameter group, you may do so by using LURA = 2.

---

3.8 **AGSETF(TPGN,FUSR)**. Allows a user program to store a floating-point number as the value of a single parameter.

3.8.1 **AGSETF - USAGE**. This subroutine transfers the value of FUSR to a local array FURA, dimensioned 1, and executes the statement

```fortran
CALL AGSETF (TPGN,FURA,1)
```

See the description of AGSETF, above. The "special actions" described there may result from a call to AGSETF.

3.8.2 **AGSETF - ARGUMENTS**. CHARACTER*(*) TPGN; REAL FUSR.

**TPGN** is an parameter identifier, as described for AGSETF, above. If a group of more than one parameter is specified, only the first element of that group will be affected by the call.

---

**ROUTINES**

---

**AUTOGRAPH**
FUSR is the floating-point value to be given to the parameter specified by TPGN.

3.9 **AGSETI (TPGN,IUSR)**. Allows a user program to store the floating-point equivalent of an integer as the value of a single parameter.

3.9.1 **AGSETI — USAGE.** This subroutine stores the value FLOAT(IUSR) in a local array FURA, dimensioned 1, and then executes the statement

CALL AGSETP (TPGN,FURA,1)

See the description of AGSETP, above. The "special actions" described there may result from a call to AGSETI.

3.9.2 **AGSETI — ARGUMENTS.** CHARACTER*(*) TPGN; INTEGER IUSR.

TPGN is a parameter identifier, as described for AGSETP, above. If a group of more than one parameter is specified, only the first element of that group will be affected by the call.

IUSR is the integer equivalent of the floating-point value to be given to the parameter specified by TPGN.

3.10 **AGSEIC (TPGN,CUSR)**. Allows a user program to (in effect) store a character string as the value of a specified single parameter.

3.10.1 **AGSEIC — USAGE.** This subroutine stores the character string CUSR in an internal string storage space, generates a floating-point identifier allowing for later retrieval of the character string, stores that identifier in a local array FURA, dimensioned 1, and then executes the statement

CALL AGSETP (TPGN,FURA,1)

See the description of AGSETP, above. The "special actions" described there may result from a call to AGSEIC.

3.10.2 **AGSEIC — ARGUMENTS.** CHARACTER*(*) TPGN, CUSR.

TPGN is a parameter identifier, as described for AGSETP, above. The specified parameter must be one of those which intrinsically have values of type character: 'LINE/END.', 'LABEL/NAME.', 'LINE/TEXT.', or 'DASH/PATTERN/n.'

CUSR is a desired character string.
If 'LINE/END.' is being set, only the first character of CUSR will be used.
If 'LABEL/NAME.' is being set, the length of CUSR will be taken to be "MAX(1,LEN(CUSR))".
If the text of a label is being set, CUSR must either be of the exact length specified by "LINE/MAXIMUM." (40 characters, by default) or shorter; if shorter, it must be terminated by the character defined by "LINE/END." (default - a '$').

If a dash pattern is being set, the length of CUSR will be taken to be the minimum of "LEN(CUSR)" and the value specified by "DASH/LENGTH."

3.11 AGGETP(TPGN,FURA,LURA). Allows a user program to get the values of a group of parameters containing one or more elements.

3.11.1 **AGGETP - USAGE.** The subroutine AGGETP is called with a character string identifying a group of parameters (possibly a single parameter), an array to receive the values of those parameters, and the length of the array, as arguments. It transfers values from the appropriate locations in the labeled common block AGCONP to the user array.

If a single parameter is being retrieved, one of the routines AGGETF, AGGETI, or AGGETC (which see, below) may be used instead. No "special" action is implied for any single parameter (as is the case for AGSETP).

See the section "PARAMETERS" for descriptions of parameters whose values may be retrieved.

3.11.2 **AGGETP - ARGUMENTS.** CHARACTER*(*) TPGN; REAL FURA(LURA); INTEGER LURA.

TPGN is a character string of the form 'k(1)/k(2)/...k(n).', where each of the k(i)'s is a keyword. The keyword k(1) specifies a group of parameters, k(2) a subgroup of that group, k(3) a subgroup of that subgroup, etc. The whole string is the name of some group of parameters the user wishes to get.

See the AGSETP argument TPGN, above, for an example and additional comments.

FURA is a user array (of length LURA) into which the floating-point values of the parameters in the group specified by TPGN are to be transferred, in the same order as they appear in the group.

See the AGSETP argument FURA, above, for additional comments.

LURA is the length of FURA. Its value may be less than, equal to, or greater than, the length of the group specified by TPGN. The number of values transferred into FURA is the minimum of the two (but not less than one). You may, for example, get the first two parameters of a 100-parameter group by using LURA = 2.

3.12 AGGETF(TPGN,FUSR). Allows a user program to retrieve the floating-point value of a single parameter.
3.12.1 **AGGETP - USAGE.** This subroutine executes the statement

```
CALL AGGETP (TPGN,FURA,1)
```

where FURA is a local array, dimensioned 1, and then sets FUSR equal to FURA(1).

See the description of AGGETP, above.

3.12.2 **AGGETP - ARGUMENTS.** CHARACTER*(*) TPGN; REAL FUSR.

TPGN is a parameter identifier, as described for AGGETP, above. If a group of more than one parameter is specified, only the first element of that group will be retrieved by the call.

FUSR receives the floating-point value of the parameter specified by TPGN.

3.13 **AGGETI(TPGN,IUSR).** Allows a user program to retrieve the integer equivalent of the floating-point value of a single parameter.

3.13.1 **AGGETI - USAGE.** This subroutine executes the statement

```
CALL AGGETP (TPGN,FURA,1)
```

where FURA is a local array, dimensioned 1, and then sets IUSR equal to IFIX(FURA(1)).

See the description of AGGETP, above.

3.13.2 **AGGETI - ARGUMENTS.** CHARACTER*(*) TPGN; INTEGER IUSR.

TPGN is a parameter identifier, as described for AGGETP, above. If a group of more than one parameter is specified, only the first element of that group will be retrieved by the call.

IUSR receives the integer equivalent of the floating-point value of the parameter specified by TPGN.

3.14 **AGGETC(TPGN,CUSR).** Allows a user program to retrieve (in effect) the character-string values of certain single parameters.

3.14.1 **AGGETC - USAGE.** This subroutine executes the statement

```
CALL AGGETP (TPGN,FURA,1)
```

where FURA is a local array, dimensioned 1. It then retrieves, from AUTOGRAPH's character storage space, the character string identified by FURA(1), and returns that string as the value of CUSR.
See the description of AGGETP, above.

3.14.2 AGGETC - ARGUMENTS. CHARACTER(*) TPGN, CUSR.

TPGN is a parameter identifier, as described for AGGETP, above. The specified parameter must be one of those which intrinsically have values of type character: 'LINE/END.', 'LABEL/NAME.', 'LINE/TEXT.', or 'DASH/PATTERN/n.'

CUSR receives the desired character string.

3.15 AGSTUP(XDRA,NVIX,IIIVX,NEVX,IIEX,...). (The remaining arguments are YDRA, NVIY, IVY, NEVY, and IIEY.) Performs "set-up" tasks required before AGBACK and AGCURV may be called. Basically, AGSTUP examines the current values of the primary control parameters for errors and computes from them and from its arguments the values of secondary control parameters. The primary and secondary control parameters together determine how the routines AGBACK and AGCURV will behave.

3.15.1 AGSTUP - USAGE. The subroutine AGSTUP is normally called once per graph, just prior to the sequence of calls to AGBACK and/or AGCURV which actually draws the graph.

The call to AGSTUP may be omitted only if (1) no primary control parameters have been changed since the last time AGSTUP was called and (2) the position of the grid window and the mapping of user x/y coordinates into the grid window would be unaffected by the AGSTUP call. The routine must be called at least once, for the first graph; for succeeding graphs, if the call can be omitted, it should be, since the routine is rather time-consuming.

Note that each of the routines EZY, EZXY, EZMY, and EZMXY unconditionally executes a call to AGSTUP (via a routine called AGEZSU) before calling AGBACK and/or AGCURV.

An appropriate call to the plot-package routine SET is executed by AGSTUP.

3.15.2 AGSTUP - ARGUMENTS. REAL XDRA(*), YDRA(*); INTEGER NVIX, IIVX, NEVX, IIEX, NVIY, IVY, NEVY, IIEY.

The first five arguments of AGSTUP are meaningful only if at least one of 'X/MINIMUM.' or 'X/MAXIMUM.' has the value "null 1" or "null 2", specifying that AUTOGRAPH is to determine for itself the minimum and/or maximum x coordinate in the user's data. Similarly, the second five arguments are meaningful only if at least one of 'Y/MINIMUM.' and 'Y/MAXIMUM.' has the value "null 1" or "null 2".

XDRA is an array of user x coordinates.

NVIX is the number of "vectors" of data from XDRA to be considered in computing the minimum and/or maximum x values.
IIVX is the index increment between two "vectors" in XDRA. The first element of the first vector is XDRA(1), the first element of the second vector is XDRA(1+IIVX), the first element of the third vector is XDRA(1+IIVX*2), etc.

NEVX is the number of elements of each vector in XDRA to be considered in computing the minimum and/or maximum x values.

IIEX is the index increment between two consecutive elements of a vector in XDRA. The second element of the first vector is XDRA(1+IIEX), the third element of the first vector is XDRA(1+IIEX*2), etc.

If IIEX has the value 0, the contents of XDRA are ignored completely; the minimum and maximum x values are considered to be "1." and FLOAT(NEVX), respectively.

YDRA, NVIX, IVY, NEVY, and IIEY are treated analogously, but define the user y coordinates.

Some examples:

<table>
<thead>
<tr>
<th>X array</th>
<th>Data to be used</th>
<th>XDRA</th>
<th>NVIX</th>
<th>IIVX</th>
<th>NEVX</th>
<th>IIEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(100)</td>
<td>(all data)</td>
<td>X(1)</td>
<td>1</td>
<td>-</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>(X(I), I=1,99,2)</td>
<td>X(1)</td>
<td>1</td>
<td>-</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>(X(I), I=51,99,2)</td>
<td>X(51)</td>
<td>1</td>
<td>-</td>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>X(10,10)</td>
<td>(all data)</td>
<td>X(1,1)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>((X(I,J), I=1,10), J=1,6)</td>
<td>X(1,1)</td>
<td>1</td>
<td>-</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X(1,1)</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X(1,1)</td>
<td>10</td>
<td>1</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X(1,1)</td>
<td>1</td>
<td>-</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>((X(I,J), I=3,7), J=3,9)</td>
<td>X(3,3)</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X(3,3)</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>((X(I,J), I=3,7,4), J=3,9,3)</td>
<td>X(3,3)</td>
<td>3</td>
<td>30</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(none)</td>
<td>1., 2., ... 62.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>62</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: The character "-" is used above to indicate an argument which is ignored and may be given a dummy value.

Normally, the x and y coordinate data tendered to AGSTUP is the same data which will later be used to draw curves. However, this need not be the case. For example, one could call AGSTUP with a two-word XDRA (setting NVIX=1, IIVX=1, NEVX=2, and IIEX=1) containing a desired minimum and maximum to be used, disregarding the real data.

If "INVERT." is given the value "1." (in place of its default value "0."), AGSTUP will behave as if its first five arguments had been interchanged with its last five, so that x-coordinate values refer to vertical distances, and y-coordinate values to horizontal distances, on the graph. This parameter affects AGCURV in a similar manner, thus allowing one to plot "x as a function

AUTOGRAPH - 37 - Routines
3.16 **AGBACK.** Draws the background specified by the current values of the control parameters - the primary parameters with default values or with values supplied by the user, and the secondary parameters with values computed by AGSTUP.

3.16.1 **AGBACK - USAGE.** Just call it. See the section "PARAMETERS" for descriptions of the parameters which affect the appearance of a background drawn by AGBACK. See also the description of AGSTUP, above.

An appropriate call to the plot-package routine SET is executed by AGBACK.

3.16.2 **AGBACK - ARGUMENTS.** None.

3.17 **AGCURV(XVEC,IIEX,YVEC,IIEY,NEXY,KDSH).** Draws a curve in a manner specified by the current values of the control parameters - the primary parameters with default values or with values supplied by the user, and the secondary parameters with values computed by AGSTUP.

3.17.1 **AGCURV - USAGE.** The subroutine AGCURV, given the x and y coordinates of a set of data points, draws the curve defined by those points, using a dashed-line pattern selected by the final argument.

See the section "PARAMETERS" for a description of the parameters which affect the behavior of AGCURV. One parameter of particular interest is 'WINDOW.' which has the default value "0." If 'WINDOW.' is given the value "1.", any portion of a curve which lies outside the grid window is omitted. No distortion of any curve segment results; the effect is exactly as if the curve were viewed through a window. There is an additional advantage in setting 'WINDOW.' to "1." if either the x coordinates, or the y coordinates, or both, are mapped logarithmically into the grid window and zero or negative values occur in the data, AGCURV treats those values as missing-point signals, rather than bombing with an ALOG10 error.

3.17.2 **AGCURV - ARGUMENTS.** REAL XVEC(*),YVEC(*); INTEGER IIEX, IIEY, NEXY, KDSH

XVEC, when IIEX is non-zero, is a singly-subscripted array containing NEXY x-coordinate data - curve point 1 has x coordinate XVEC(1), curve point 2 has x coordinate XVEC(1+IIEX), curve point 3 has x coordinate XVEC(1+IIEX*2), etc. When IIEX is zero, the array XVEC is ignored - curve point 1 has x coordinate "1.", curve point 2 has x coordinate "2.", etc.

If the value of any x coordinate matches the current value of 'NULL/1.' (default - "1.E36"), the corresponding point is considered to be missing - curve segments on either side of that point are not drawn.
IIEX, if non-zero, is the index increment between one x coordinate in XVEC and the next. If IIEX is zero, the array XVEC is ignored, as described above.

YVEC is just like XVEC, but provides y coordinate data.

IIEY is just like IIEX, but describes the use (or non-use) of YVEC.

NE=NX is the number of curve points – the number of x/y coordinate pairs to be used.

Note: If "INVERT." is given the value "1." (in place of its default value "0."), AGCURV will behave as if the arguments XVEC and IIEX had been interchanged with the arguments YVEC and IIEY, so that x-coordinate values refer to vertical distances, and y-coordinate values to horizontal distances, on the graph. This parameter affects AGSTUP in a similar manner, thus allowing one to plot "x as a function of y".

KDSH specifies the dashed-line pattern to be used in drawing the curve.
(Since the routines DASHD, FRSTD, VECTD, and LASTD, in the package DASHCHAR, are used to draw the curve, it may have its own particular dashed-line pattern.)

If KDSH is zero, the user is assumed to have done his own call to DASHD; AGCURV will do no such call.

If KDSH is non-zero and negative, the function MOD(-KDSH-1,26)+1 determines which of 26 "alphabetic" patterns is to be used; each of these generates a solid line interrupted by one of the letters of the alphabet. The value 1 implies that an "A" will be used, the value 2 that a "B" will be used, ... the value 27 that an "A" will be used again, etc.

If KDSH is non-zero and positive, the function MOD(KDSH-1,n)+1 determines which of n "user" patterns is to be used; these n patterns are defined by the parameters in the group named "DASH." - the default values specify one solid-line pattern.

Note: The routines EZY and EZXY, which draw one curve per call, always call AGCURV with KDSH = 1. The routines EZMY and EZMXY, which draw one or more curves per call, call AGCURV with KDSH = ISIGN(p,q), where p is the number of the curve being drawn (p is between 1 and MANY, inclusive) and q is the current integral value of "DASH/SELECTOR.".

3.18 AGSAVE(IFNO). Saves the current state of AUTOGRAPH for later restoration by AGRSTR.

3.18.1 AGSAVE - USAGE. Calling AGSAVE saves the current state of AUTOGRAPH (frequently the default state) by writing, on the unit specified by IFNO, the current values of all the parameters and the contents of the character storage space referenced by certain of those parameters.
3.18.2 AGSAVE - 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. AGSAVE does not rewind it, either before or after writing the record.

3.19 AGRSTR(IFNO). Restores a saved state of AUTOGRAPH.

3.19.1 AGRSTR - USAGE. Calling AGRSTR restores AUTOGRAPH to a previously saved state (frequently the default state) by reading, from the unit specified by IFNO, values of all the parameters and the contents of the character storage space referenced by certain of those parameters.

3.19.2 AGRSTR - 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. AGRSTR does not rewind it, either before or after reading the record.

3.20 AGBNCH(IDSH). Provides an easy way to convert binary dash patterns into character dash patterns.

3.20.1 AGBNCH - USAGE. AGBNCH is a function, of type CHARACTER*16, and must be declared as such in a user program referencing it:

```
CHARACTER*16 AGBNCH
```

The value of AGBNCH(IDSH), where IDSH is an integer in the range 0 to 65535 (2**16-1) representing a 16-bit binary dash pattern, is the equivalent character dash pattern.

3.20.2 AGBNCH - ARGUMENTS. INTEGER IDSH.

IDSH is an integer in the range 0 to 65535, inclusive.

3.21 AGDSHN(IDSH). Provides an easy way to generate the names of parameters in the group 'DASH/PATTERN:', for use in calls to AGSETC and AGGETC.

3.21.1 AGDSHN - USAGE. AGDSHN is a function, of type CHARACTER*16, and must be declared as such in a user program referencing it:

```
CHARACTER*16 AGDSHN
```

The value of AGDSHN(IDSH), where IDSH is an integer "n" in the range 1 to 26, is the name of the nth dash-pattern parameter - that is to say, it is the character string 'DASH/PATTERN/n'.
3.21.2 HN - ARGUMENTS. INTEGER IDSH.

IDSH is an integer in the range 1 to 26, inclusive.

3.22 AGOTOL(IAxSFuNs,VI ,,vINPW ). Provides a way for the user to change the user-system-to-label-system mapping for one or more of the four axes.

3.22.1 AGOTOL - USAGE. This routine is not called by the user program, but by AUTOGRAPH itself. It defines the user-system-to-label-system mapping for all four axes. The default version makes the label system match the user system on all four axes. The user may supply his own version to change the mapping on one or more of the axes. Mappings defined by the subroutine must be continuous and monotonic.

Note: A user version of AGOTOL should not call any other AUTOGRAPH routine.

3.22.2 AGOTOL - ARGUMENTS. INTEGER IAXS, IDMA; REAL FUNS, VINP, VOTP.

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 "AXIS/s/FUNCTION.", which may be used to select the desired mapping function for the axis IAXS. It is recommended that the default value (zero) be used to specify the identity mapping. The value may be integral ("1.", "2.", etc.) and serve purely to select the code to be executed or it may be the value of an actual 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. The mappings in one direction must be the mathematical inverses of the mappings in the other direction or AUTOGRAPH will probably go bananas.

VINP is an input value in one coordinate system.

VOTP is an output value in the other coordinate system.

3.23 AGCHAX(IFLG, IAXS, IPRT, VILS). Provides a way for the user to change the color, intensity, line style, etc., of various portions of the axes.

3.23.1 AGCHAX - USAGE. This routine is not called by the user program, but by AUTOGRAPH itself, just before and just after each of the objects making up an axis is drawn. The default version does nothing. The user may supply his own version to change the color, intensity, line style, etc. of selected portions of the axis.

Note: A user version of AGCHAX should not call any other AUTOGRAPH routine.
3.23.2 ACAX - ARGUMENTS. INTEGER IFLG, IAXS, IPRT; REAL VILS.

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.

3.24 AGCHCU(IFLG,KDSH). Provides a way for the user to change the color, intensity, line style, etc., of curves drawn by AUTOGRAPH.

3.24.1 AGCHCU - USAGE. This routine is not called by the user program, but by AUTOGRAPH itself, just before and just after each curve is drawn. The default version does nothing. The user may supply his own version to change the color, intensity, line style, etc. of selected curves.

Note: A user version of AGCHCU should not call any other AUTOGRAPH routine.

3.24.2 AGCHCU - ARGUMENTS. INTEGER IFLG, KDSH.

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

KDSH is the value of AGCURV's last argument, 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;; n is the curve number</td>
</tr>
<tr>
<td>EZMXY</td>
<td>&quot;n&quot; or &quot;-n&quot;; n is the curve number</td>
</tr>
<tr>
<td>the user program</td>
<td>the user value</td>
</tr>
</tbody>
</table>
3.25 **AGCHIL(IFG, LBNM, LNNO)**. Provides a way for the user to change the color, intensity, text style, etc., of the informational labels.

3.25.1 **AGCHIL - USAGE.** This routine is not called by the user program, but by AUTOGRAPH itself, just before and just after each informational-label line is drawn. The default version does nothing. The user may supply his own version to change the appearance of selected lines of text.

Note: A user version of AGCHIL should not call any other AUTOGRAPH routine.

3.25.2 **AGCHIL - ARGUMENTS.** INTEGER IFG, LNNO; CHARACTER* (*) LBNM.

IFG 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.

---------------------------------------------------

3.26 **AGCHNL(IAXS, VILS, CHRM, MCIM, NCIM, ...)**. (The remaining arguments are IPXM, CHRE, MCIE, and NCIE.) Provides a way for the user to substitute arbitrary character strings for the numeric labels generated by AUTOGRAPH.

3.26.1 **AGCHNL - USAGE.** This routine is not called by the user program, but by AUTOGRAPH itself, just after the character string representing each numeric label has been generated and just before it is written on the graph. The user may change the character string in any desired way. Axes may thereby be labeled using the names of the months, Roman numerals, etc.

Note: A user version of AGCHNL should not call any other AUTOGRAPH routine.

3.26.2 **AGCHNL - ARGUMENTS.** INTEGER IAXS, MCIM, NCIM, IPXM, MCIE, NCIE; REAL VILS; CHARACTER* (*) CHRM, CHRE.

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.

NCIE is the length of CH(RE - the maximum number of characters that it will hold. The value of NCIE must not be altered.

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

3.27 AGPWRT(XPOS,YPOS,CHRS,NCHS,ISIZ,IORI,ICEN). Provides a way for the user to change the style of all text strings drawn by AUTOGRAPH.

3.27.1 AGPWRT - USAGE. This routine is not called by the user program, but by AUTOGRAPH itself, to draw a numeric label or an informational label on the graph. The default version just calls the plot-package routine PWRIT. The user may supply a replacement version. For example, the XLIB file AGUPWRITX contains a version of AGPWRT which calls PWRITX, which produces much nicer-looking labeling.

Note: A user version of AGPWRT should not call any other AUTOGRAPH routine.

3.27.2 AGPWRT - ARGUMENTS. REAL XPOS, YPOS; CHARACTER*(*) CHRS; INTEGER NCHS, ISIZ, IORI, ICEN.

XPOS is the x coordinate of a point relative to which the text string is to be positioned.

YPOS is the y coordinate of a point relative to which the text string is to be positioned.

CHRS is the text string to be drawn.

NCHS is the length of CHRS - the number of characters in the text string.

ISIZ specifies the size of the characters to be used. The values 1, 2, 3, and 4 imply character widths of 8, 12, 16, and 24 plotter units, respectively. Larger values specify the character width directly.

IORI is the orientation angle of the text string, measured in degrees counterclockwise from a vector which is horizontal and pointing to the right.

ICEN is the centering option. The value "-1" implies that the text is to be written with (XPOS,YPOS) in the center of the left edge of the leftmost character, the value "0" that (XPOS,YPOS) is to be in the center of the entire string, and the value "+1" that (XPOS,YPOS) is to be in the center of the right edge of the rightmost character.
4. PARAMETERS

The AUTOGRAPH control parameters reside in the labeled common block AGCONP. There are currently 485 of them, of which 336 are "primary" and 149 are "secondary". Primary control parameters have default values and are subject to change by a user program to produce some desired effect on the behavior of AUTOGRAPH and/or on the nature of a graph being drawn. Secondary control parameters are computed by AUTOGRAPH itself and are not normally subject to change by a user program.

User access to these parameters is provided by the routines AGSETP, AGSETF, AGSETI, AGSETC, AGGETP, AGGETF, AGGETI, and AGGETC (all of which are described in the section "ROUTINES", above). The first argument in a call to one of these routines is a character string naming a group of parameters (perhaps containing only a single parameter) which the user wishes to "set" or "get". Each such string has the form

```
'k(l)/k(2)/...k(n).'
```

where k(l) is a keyword identifying a major group of parameters, k(2) is a keyword identifying a subgroup of that major group, k(3) is a keyword identifying a further subgroup of that subgroup, and so on. Only the first three characters of k(l) and the first two characters of the others need be used; also, certain keywords may be omitted.

Because of certain portability considerations, all of the parameters have floating-point values. The routine AGSETP may be used to set the floating-point values of the parameters in any group, the routine AGGETP to retrieve those values. The routine AGSETF (AGGETF) may be used to set (get) the floating-point value of any single parameter.

Some parameters may only take on discrete integral values (like "0.", "1.", "-6.", or "65535.") and are used in roles for which integers would normally be used. The routine AGSETI (AGGETI) may be used to set (get) the integer value of any single parameter of this type.

Other parameters intrinsically represent character strings; the floating-point value of the parameter is an identifier, generated when the character string is passed to AUTOGRAPH and enabling the character string to be retrieved from AUTOGRAPH's character storage space when it is needed. The routine AGSETC (AGGETC) must be used to set (get) the character-string value associated with any single parameter of this type.

Many parameters have a limited range of acceptable values. What generally happens when a parameter is given an out-of-range value is that AUTOGRAPH (usually the routine AGSTUP) resets that value to the value at the nearer end of the acceptable range.

Setting certain parameters (individually, rather than as part of a multi-parameter group) implies, as a side effect, "special action" by the routine AGSETP (which may be called directly by the user or indirectly by way of a user call to AGSETF, AGSETI, or AGSETC). For example, setting the parameter "BACKGROUND." to request a particular background type causes a number of other
parameters to be changed in order to achieve the desired result.

Each of the named parameter groups is described below. Square brackets are used to mark portions of a name which may be omitted; the notation

\[ k(1)/k(2)/...[k(i)]...k(n). \]

indicates that the keyword \( k(i) \) and the following slash may be omitted. In each description, the simplest form of the name is given. If a multi-parameter group is named, its subgroups are listed, in the order in which they occur in the group. If a single parameter is named, the default value of that parameter is given and any "special action" by AGSETP is described.

4.1 'PRIMARY.'

Simplest form of name: 'PRI.'

This group consists of all 336 primary control parameters, in the order in which they appear below. It was originally provided to give users the capability of saving and restoring the state of AUTOGRAPH. The routines AGSAVE and AGRSTR (which see) should now be used for this purpose.

4.2 'FRAME.'

Simplest form of name: 'FRA.'

An integral floating-point number specifying when a frame advance is to be done by the routines EZY, EZXY, EZMY, and EZMXY and having one of three possible values:

- The value "1." specifies a frame advance after drawing a graph.
- The value "2." specifies no frame advance at all.
- The value "3." specifies a frame advance before drawing a graph.

Default value: "1." (frame advance after drawing graph).

4.3 'SET.'

Simplest form of name: 'SET.'

An integral floating-point number specifying whether or not the arguments of the last call to "SET" are used to determine the linear/log nature of the current graph, the position of the grid window and/or the x/y minimum/maximum values.
(Note: The routine SET is part of the system plot package. Its first four arguments specify a portion of the plotter frame, its next four arguments specify the minimum and maximum x and y coordinate values to be mapped to that portion, and its ninth argument specifies the linear/log nature of the mapping. The routine GETSET, which is also a part of the system plot package, is used to retrieve the arguments of the last call to SET.)

Giving SET a value (individually, rather than as part of a group) has both an immediate effect and a delayed effect. The immediate effect, which occurs in the routine AGSETP, is to return most of the parameters in the groups GRID, X, and Y to their default values. (Exceptions are X/LOGARITHMIC and Y/LOGARITHMIC, which may have values making them immune to such resetting.) The delayed effect, which occurs in the routine AGSTUP, depends on the value given to SET.

There are eight acceptable values of SET, four of which are just the negatives of the other four. Using a negated value suppresses the drawing of curves by the routines EZY, EZXY, EZMY, and EZMXY. Acceptable absolute values of SET are as follows:

- The value "1." means that the arguments of the last SET call are not to be used by AGSTUP.
- The value "2." means that X/LOGARITHMIC and Y/LOGARITHMIC are to be given values ("0." or "-1.") consistent with the ninth argument of the last SET call and that parameters in the group GRID are to be given values consistent with the first four arguments of the last SET call.
- The value "3." means that X/LOGARITHMIC and Y/LOGARITHMIC are to be given values ("0." or "-1.") consistent with the ninth argument of the last SET call and that the other parameters in the groups X and Y are to be given values consistent with the fifth through eighth arguments of the last SET call.
- The value "4." means a combination of the actions specified by the values "2." and "3."

Default value: "1." (no arguments of last SET call used).

Special action by AGSETP: As described above, if SET is set (individually, rather than as part of a group) to any value by an AGSETP call, the parameters in the groups GRID, X, and Y are reset to their default values. The parameter X/LOGARITHMIC is reset to its default value ("0.") only if it has the value "+1."; a value of "-1." is not changed; Y/LOGARITHMIC is treated similarly.

4.4 'ROW'.

Simplest form of name: 'ROW'.
An integral floating-point number specifying the assumed dimensioning of x and y coordinate data arrays used in calls to the routines EZMY and EZMXY. There are four possibilities:

- The value "-2." means that both x and y arrays are subscripted by curve number and point number, in that order.

- The value "-1." means that y arrays are subscripted by curve number and point number, in that order, but that x arrays are subscripted by point number only. (The same x-coordinate data is used for all the curves.)

- Either of the values "0." or "1." means that y arrays are subscripted by point number and curve number, in that order, but that x arrays are subscripted by point number only. (The same x-coordinate data is used for all of the curves.)

- The value "+2." means that both x and y arrays are subscripted by point number and curve number, in that order.

Default value: "1." (y by point and curve numbers, x by point number only).

---

4.5 "INVERT".

Simplest form of name: 'INV.'

An integral floating-point number having the value "0." or "1."; giving it the value "1." causes the routines AGSTUP and AGCURV to behave as if arguments defining x-coordinate data had been interchanged with arguments defining y-coordinate data, thus, in some sense, allowing one to graph "x as a function of y". This parameter is principally intended for users of the routines EZY, EZXY, EZMY, and EZMXY.

Default value: "0." (no inversion of x and y arguments).

---

4.6 "WINDOW".

Simplest form of name: 'WIN.'

An integral floating-point number having the value "0." or "1."; giving it the value "1." causes the routine AGCURV to use the subroutine AGQURV, rather than AGKURV, for drawing curves. The result is that curve portions falling outside the grid window are omitted. See the description of the routine AGCURV, above.

Default value: "0." (no windowing of curves).
4.7 `'BACKGROUND.'`

Simplest form of name: `'BAC.'`

An integral floating-point number specifying the type of background to be drawn by `AGBACK`. There are four acceptable values:

- The value "1." specifies a "perimeter" background.
- The value "2." specifies a "grid" background.
- The value "3." specifies a "half-axis" background.
- The value "4." specifies no background at all.

Default value: "1." (a "perimeter" background).

Special action by `AGSETP`: If `'BACKGROUND.'` is set (individually, rather than as part of a group) by a call to `AGSETP`, the desired background is created by changing the following parameters:

```plaintext
'[AXIS/]s/CONTROL.'
'[AXIS/]s/[TICKS/]MAJOR/[LENGTH/]INWARD.'
'[AXIS/]s/[TICKS/]MINOR/[LENGTH/]INWARD.'
'LABEL/CONTROL.'
```

where "s" stands for "LEFT", "RIGHT", "BOTTOM", and "TOP". This determines which of the axes are plotted, how long the inward-pointing portions of major and minor tick marks are to be, and whether or not informational labels are to be plotted. Values used are as follows:

The value "1." (perimeter background) sets:

```plaintext
's/CONTROL.' to "4." for all s;
's/MAJOR/INWARD.' to ".015" for all s;
's/MINOR/INWARD.' to ".010" for all s;
'LABEL/CONTROL.' to "2.".
```

The value "2." (grid background) sets:

```plaintext
's/CONTROL.' to "4." for "s" = "LEFT" and "BOTTOM",
's/CONTROL.' to ":-1." for "s" = "RIGHT" and "TOP";
's/MAJOR/INWARD.' to "1." for all s;
's/MINOR/INWARD.' to "1." for all s;
'LABEL/CONTROL.' to "2.".
```

The value "3." (half-axis background) sets:

```plaintext
's/CONTROL.' to "4." for "s" = "LEFT" and "BOTTOM",
's/CONTROL.' to ":-1." for "s" = "RIGHT" and "TOP";
's/MAJOR/INWARD.' to ".015" for all s;
's/MINOR/INWARD.' to ".010" for all s;
'LABEL/CONTROL.' to "2.".
```
The value "4." (no background) sets:

- "s/CONTROL." to "0." for all s;
- "s/MAJOR/INWARD." to ".015" for all s;
- "s/MINOR/INWARD." to ".010" for all s;
- "LABEL/CONTROL." to "0.".

The default values of these thirteen parameters correspond to the default value of "BACKGROUND.". Note that, if they are changed directly, the value of "BACKGROUND." may not reflect the actual nature of the background defined by them.

4.8 "NULL.".

Simplest form of name: "NULL."

This group contains the two "nulls" (or "special values") "NULL/1." and "NULL/2.".

4.9 "NULL/1.".

Simplest form of name: "NULL/1."

A floating-point number "null 1", used in the following ways by AUTOGRAPH:

- Certain parameters have by default, or may be given, the value "null 1", specifying that the routine AGSTUP is to choose values for them. The value chosen for a given parameter is not back-stored in place of the "null 1"; thus, a unique value will be chosen for each graph drawn.

- If a curve point specified by the user has x and/or y coordinates equal to "null 1", that curve point is ignored. It is not used in computing minimum and maximum values. Curve segments on either side of it are not drawn.

Default value: "1.E36" (an arbitrary value).

Special action by AGSETP: If "NULL/1." is changed (individually, rather than as part of a group) by an AGSETP call, the entire list of primary parameters is scanned - any value equal to the old "null 1" is replaced by the new one.

4.10 "NULL/2.".

Simplest form of name: "NULL/2."

A floating-point number "null 2". Certain parameters may be given the value "null 2", specifying that the routine AGSTUP is to choose values for them.
The value chosen for a given parameter is back-stored in place of the "null 2"; thus, a unique value may be chosen for the first graph of a series and then used for all remaining graphs in the series.

Default value: "2.E36" (an arbitrary value).

Special action by AGSETP: If 'NULL/2.' is changed (individually, rather than as part of a group) by an AGSETP call, the entire list of primary parameters is scanned - any value equal to the old "null 2" is replaced by the new one.

4.11 'GRAPH.'

Simplest form of name: 'GRA.'

A group of four parameters describing the position of the "graph window" within the plotter frame. A graph drawn by AUTOGRAPH (including labels) is forced to lie entirely within this window. Subgroups and the number of parameters in each are as follows:

'GRAPH/LEFT.' (1)
'GRAPH/RIGHT.' (1)
'GRAPH/BOTTOM.' (1)
'GRAPH/TOP.' (1)

4.12 'GRAPH/LEFT.'

Simplest form of name: 'GRA/LE.'

A floating-point number between "0." and "1." specifying the position of the left edge of the graph window as a fraction of the distance from the left edge to the right edge of the plotter frame.

Default value: "0." (left edge of plotter frame).

4.13 'GRAPH/RIGHT.'

Simplest form of name: 'GRA/RI.'

A floating-point number between "0." and "1." specifying the position of the right edge of the graph window as a fraction of the distance from the left edge to the right edge of the plotter frame.

Default value: "1." (right edge of plotter frame).
4.14 'GRAPH/BOTTOM.'.

Simplest form of name: 'GRA/BO.'

A floating-point number between "0." and "1." specifying the position of the bottom edge of the graph window as a fraction of the distance from the bottom edge to the top edge of the plotter frame.

Default value: "0." (bottom edge of plotter frame).

4.15 'GRAPH/TOP.'.

Simplest form of name: 'GRA/TO.'

A floating-point number between "0." and "1." specifying the position of the top edge of the graph window as a fraction of the distance from the bottom edge to the top edge of the plotter frame.

Default value: "0." (top edge of plotter frame).

4.16 'GRID.'.

Simplest form of name: 'GRI.'

A group of five parameters describing the position and shape of the "grid window" within the graph window. Subgroups and the number of parameters in each are as follows:

'GRID/LEFT.' (1)
'GRID/RIGHT.' (1)
'GRID/BOTTOM.' (1)
'GRID/TOP.' (1)
'GRID/SHAPE.' (1)

4.17 'GRID/LEFT.'.

Simplest form of name: 'GRI/LE.'

A floating-point number between "0." and "1." specifying the position of the left edge of the area in which the grid window is to be placed, stated as a fraction of the distance from the left edge to the right edge of the graph window.

Default value: ".15".
4.18 'GRID/RIGHT.'.

Simplest form of name: 'GRI/RI.'

A floating-point number between "0." and "1." specifying the position of the right edge of the area in which the grid window is to be placed, stated as a fraction of the distance from the left edge to the right edge of the graph window.

Default value: ".95".

4.19 'GRID/BOTTOM.'.

Simplest form of name: 'GRI/BO.'

A floating-point number between "0." and "1." specifying the position of the bottom edge of the area in which the grid window is to be placed, stated as a fraction of the distance from the bottom edge to the top edge of the graph window.

Default value: ".15".

4.20 'GRID/TOP.'.

Simplest form of name: 'GRI/TO.'

A floating-point number between "0." and "1." specifying the position of the top edge of the area in which the grid window is to be placed, stated as a fraction of the distance from the bottom edge to the top edge of the graph window.

Default value: ".95".

4.21 'GRID/SHAPE.'.

Simplest form of name: 'GRI/SH.'

A floating-point number specifying the shape of the grid window. The grid window, whatever its shape, is centered in, and made as large as possible in, the area specified by the first four parameters in the group 'GRID.'. The value of 'GRID/SHAPE.' falls in one of four possible ranges, as follows:

- A value less than "0." specifies the negative of the desired ratio of the grid window's width to its height. For example, the value "-2." specifies a grid window which is twice as wide as it is high.
The value "0." specifies a grid window of exactly the same shape as the area specified by the first four parameters in the group "GRID.". The grid window therefore fills that area completely.

A value "s" between "0." and "1." specifies a grid window whose shape is determined by the range of the user's coordinate data, reverting to the shape of the area specified by the first four arguments in the group "GRID." if the ratio of the shorter side of the grid window to the longer side of the grid window would thereby be made less than "s". For example, if "s" were given the value ".5" and the user x coordinate data ranged in value from "0." to "10." and the user y coordinate data ranged in value from "0." to "15."., the grid window would be made two-thirds as wide as it was high; however, if the y coordinate data ranged in value from "0." to "100.", the grid window would not be made one-tenth as wide as it is high, but would instead be made to fill the entire area specified by the first four arguments of the group "GRID.".

A value "s" greater than or equal to "1." specifies a grid window whose shape is determined by the range of the user's coordinate data, reverting to a square if the ratio of the longer side of the grid window to the shorter side of the grid window would thereby be made greater than "s".

Note that, if 'GRID/SHAPE.' is given a value greater than "0.", AUTOGRAPH assumes that the user's x and y coordinate data have the same units (both in inches, for example) and that the outline of a real two-dimensional object is to be graphed without distortion. The grid window is shaped in such a way as to accomplish this. This feature should not be used when either 'X/LOGARITHMIC.' or 'Y/LOGARITHMIC.' has a non-zero value; doing so will yield strange results.

Note that either of the values "-1." or "+1." produces a square and that the value "-1.6180398874989" produces a golden rectangle.

Default value: "0.".

4.22 "X."

Simplest form of name: "X."

A group of seven parameters specifying the mapping of the user's x-coordinate data onto the horizontal axis of the grid window. Subgroups and the number of parameters in each are as follows:

"X/MINIMUM." (1)
"X/MAXIMUM." (1)
"X/LOGARITHMIC." (1)
"X/ORDER." (1)
"X/NICE." (1)
"X/SMALLEST." (1)
"X/LARGEST." (1)
See also 'SET.' and 'INVERT.', above.

4.23 'X/MINIMUM.'

Simplest form of name: 'X/MI.'

A floating-point number specifying the minimum user x coordinate to be considered. This parameter normally has the value "null 1", specifying that the routine AGSTUP should examine the user's x-coordinate data and find the minimum value for itself.

If the value "null 2" is used, it will be replaced, the next time AGSTUP is called, by an actual minimum value computed by AGSTUP.

If a non-null value is used, AGSTUP will not examine the user's x-coordinate data; the given value will be considered to be the minimum.

If both 'X/MINIMUM.' and 'X/MAXIMUM.' are given non-null values, the former should have a lesser value than the latter.

Default value: "1.E36" ("null 1").

4.24 'X/MAXIMUM.'

Simplest form of name: 'X/MA.'

Analogous to 'X/MINIMUM.', above; it specifies the way in which the maximum x coordinate is to be determined.

Default value: "1.E36" ("null 1").

4.25 'X/LOGARITHMIC.'

Simplest form of name: 'X/LO.'

An integral floating-point number having one of the values "-1.", "0.", or "+1.".

- The value "0." specifies that the mapping of user x coordinates onto the horizontal axis of the grid window is to be linear.

- The values "-1." and "+1." specify that the mapping is to be logarithmic, in which case all user x-coordinate data must be greater than zero. The value "-1." is immune to change when 'SET.' (which see, above) is reset; the value "+1." is not.
Default value: "0." (linear x mapping).

4.26 ‘X/ORDER.’

Simplest form of name: ‘X/OR.’

An integral floating-point number having one of the values "0." or "1."

- The value "0." specifies that the values of user x coordinates mapped to the horizontal axis of the grid window should increase from left to right.
- The value "1." specifies that user x coordinates should decrease from left to right.

Default value: "0." (increase from left to right).

4.27 ‘X/NICE.’

Simplest form of name: ‘X/NI.’

An integral floating-point number having one of the values "-1.", "0.", or "+1.".

- The value "-1." specifies that user x-coordinate data are to be mapped onto the horizontal axis of the grid window in such a way as to force major-tick positions at the endpoints of the bottom x axis.
- The value "+1." specifies that user x-coordinate data are to be mapped onto the horizontal axis of the grid window in such a way as to force major-tick positions at the endpoints of the top x axis.
- The value "0." specifies that the x-coordinate data are to be mapped so as to range from the left edge of the grid window to the right edge of the grid window; major-tick positions are not forced at the ends of either x axis.

Default value: "-1." (bottom axis "nice").

4.28 ‘X/SMALLEST.’

Simplest form of name: ‘X/SM.’

This parameter comes into play when AGSTUP is called upon to compute the minimum x coordinate (when ‘X/MINIMUM.’ has a null value); if the value of ‘X/SMALLEST.’ is non-null, values less than it will not be considered in the computation.
4.29 "X/LARGEST."

Simplest form of name: "X/LA."

This parameter comes into play when AGSTUP is called upon to compute the maximum x coordinate (when "X/MAXIMUM." has a null value); if the value of "X/LARGEST." is non-null, values greater than it will not be considered in the computation.

Default value: "1.E36" ("null 1").

4.30 "Y."

Simplest form of name: "Y."

A group of seven parameters specifying the mapping of the user's y-coordinate data onto the vertical axis of the grid window. Subgroups and the number of parameters in each are as follows:

- "Y/MINIMUM." (1)
- "Y/MAXIMUM." (1)
- "Y/LOGARITHMIC." (1)
- "Y/ORDER." (1)
- "Y/NICE." (1)
- "Y/SMALLEST." (1)
- "Y/LARGEST." (1)

See also "SET." and "INVERT.", above.

4.31 "Y/MINIMUM."

Simplest form of name: "Y/MI."

Analogous to "X/MINIMUM.", above; it specifies the way in which the minimum y coordinate is to be determined.

Default value: "1.E36" ("null 1").

4.32 "Y/MAXIMUM.".

Simplest form of name: "Y/MA."
Analogous to `X/MAXIMUM.', above; it specifies the way in which the maximum y coordinate is to be determined.

Default value: "1.E36" ("null 1").

---------------------------------------------------------------

4.33 `Y/LOGARITHMIC.'.

Simplest form of name: `Y/LO.'

Analogous to `X/LOGARITHMIC.', above; it specifies whether the mapping of y coordinates is linear or logarithmic.

Default value: "0." (linear y).

---------------------------------------------------------------

4.34 `Y/ORDER.'.

Simplest form of name: `Y/OR.'

Analogous to `X/ORDER.', above; it specifies whether y-coordinates increase or decrease from bottom to top.

Default value: "0." (increase from bottom to top).

---------------------------------------------------------------

4.35 `Y/NICE.'.

Simplest form of name: `Y/NI.'

Analogous to `X/NICE.', above; it specifies whether the left y axis, the right y axis, or neither, is to be "nice".

Default value: "-1." (left axis "nice").

---------------------------------------------------------------

4.36 `Y/SMALLEST.'.

Simplest form of name: `Y/SM.'

Analogous to `X/SMALLEST.', above; comes into play when AGSTUP is called upon to compute the minimum y coordinate.

Default value: "1.E36" ("null 1").

---------------------------------------------------------------
4.37 'Y/LARGEST.'

Simplest form of name: 'Y/LA.'

Analogous to 'X/LARGEST.', above; comes into play when AGSTUP is called upon to compute the maximum y coordinate.

Default value: "1.E36" ("null 1").

4.38 'AXIS.'

Simplest form of name: 'AXI.'

A group of 92 parameters describing four axes: the left axis, the right axis, the bottom axis, and the top axis. Subgroups and the number of parameters in each are as follows:

'[AXIS/] LEFT.' (23)
'[AXIS/] RIGHT.' (23)
'[AXIS/] BOTTOM.' (23)
'[AXIS/] TOP.' (23)

The elements of the subgroups are interleaved in the group; that is to say, the first elements of the four subgroups constitute elements 1 through 4 of the group, the second elements of the four subgroups constitute elements 5 through 8 of the group, and so on.

4.39 '[AXIS/]s.'

(where "s" means "any one of the keywords LEFT, RIGHT, BOTTOM, or TOP").

Simplest form of name: 's.'

A group of 23 parameters describing the axis specified by "s". Subgroups and the number of parameters in each are as follows:

'[AXIS/]s/CONTROL.' (1)
'[AXIS/]s/LINE.' (1)
'[AXIS/]s/INTERSECTION.' (2)
'[AXIS/]s/FUNCTION.' (1)
'[AXIS/]s/TICKS.' (10)
'[AXIS/]s/NUMERIC.' (8)

4.40 '[AXIS/]s/CONTROL.'
An integral floating-point number having one of the values "-1.", "0.", "1.", "2.", "3.", or "4." and controlling certain aspects of the drawing of the axis specified by "s", as follows:

- The value "-1." specifies that only the line portion of the axis may be drawn; tick marks and numeric labels are suppressed.
- The value "0." specifies that no portion of the axis may be drawn.
- A positive value specifies that all portions of the axis may be drawn and specifies what actions AUTOGRAPH may take to prevent numeric-label overlap problems, as follows:
  - The value "1." specifies that numeric labels may not be shrunk or rotated.
  - The value "2." specifies that numeric labels may be shrunk, but not rotated.
  - The value "3." specifies that numeric labels may be rotated, but not shrunk.
  - The value "4." specifies that numeric labels may be both shrunk and/or rotated.

Default value: "4." for all "s" (all axes drawn, numeric labels may be shrunk and/or rotated).

4.41 `[AXIS/]s/LINE.`

Simplest form of name: `s/LI.`

An integral floating-point number having one of the values "0." or "1.".

- The value "0." specifies that the line portion of the axis specified by "s" may be drawn.
- The value "1." suppresses the line portion of the axis specified by "s".

Default value: "0." for all "s" (line portions of all axes may be drawn).

4.42 `[AXIS/]s/INTERSECTION.`

Simplest form of name: `s/IN.`

A group of two parameters

PARAMETERS - 60 - AUTOGRAPH
each having the default value "1.E36" ("null 1"). Giving either of them a non-null value causes the axis specified by "s" to be moved away from its normal position on one edge of the grid window. If both are given non-null values, ".../USER." takes precedence over ".../GRID."

If the left y axis or the right y axis is moved, it remains vertical, but intersects the bottom of the grid window at a specified x coordinate. Similarly, if the bottom x axis or the top x axis is moved, it remains horizontal, but intersects the left edge of the grid at a specified y coordinate.

No axis may be moved outside the current graph window; if an attempt is made to do so, the axis is moved as far as the edge and no farther.

4.43 "[AXIS/] s/INTERSECTION/GRID."

Simplest form of name: "s/IN/GR."

A floating-point number which, if not equal to the current "null 1", specifies, in the grid coordinate system, the x coordinate (if "s" = "LEFT" or "RIGHT") or the y coordinate (if "s" = "BOTTOM" or "TOP") of the point of intersection of the axis specified by "s" with the perpendicular sides of the grid window.

Default value: "1.E36" ("null 1") for all "s" (axes lie on the edges of the grid window).

4.44 "[AXIS/] s/INTERSECTION/USER."

Simplest form of name: "s/IN/US."

A floating-point number which, if not equal to the current "null 1", specifies, in the user coordinate system, the x coordinate (if "s" = "LEFT" or "RIGHT") or the y coordinate (if "s" = "BOTTOM" or "TOP") of the point of intersection of the axis specified by "s" with the perpendicular sides of the grid window.

Default value: "1.E36" ("null 1") for all "s" (axes lie on the edges of the grid window).

4.45 "[AXIS/] s/FUNCTION."

Simplest form of name: "s/FU."
A floating-point number, passed as an argument to the subroutine AGUTOL; this subroutine defines the user-system-to-label-system mappings, and thus the label coordinate systems, for all the axes. The default version of AGUTOL defines the identity mapping for all axes; a user version may be substituted to define any desired set of mappings. It is intended that "AXIS/s/FUNCTION." be used within a user version of AGUTOL as a function selector. It is further recommended that the value "0." select the identity mapping, thus providing a way to re-create the default situation.

Tick marks on the axis specified by "s" are positioned in the label coordinate system. Numeric labels on the axis give values in the label coordinate system.

See the description of AGUTOL in the section "ROUTINES"; see also example 7.

Default value: "0." for all "s" (identity mapping for all axes).

-----------------------------------------------

4.46 "[AXIS/]s/TICKS.".

Simplest form of name: 's/TI.'

A group of ten parameters describing the tick marks, if any, which are to be a part of the axis specified by "s". Subgroups and the number of parameters in each are as follows:

'[AXIS/]s/[TICKS/]MAJOR.' (6)
'[AXIS/]s/[TICKS/]MINOR.' (4)

-----------------------------------------------

4.47 "[AXIS/]s/[TICKS/]MAJOR.".

Simplest form of name: 's/MA.'

A group of six parameters describing the major tick marks, if any, which are to be a part of the axis specified by "s". Subgroups and the number of parameters in each are as follows:

'[AXIS/]s/[TICKS/]MAJOR/SPACING.' (3)
'[AXIS/]s/[TICKS/]MAJOR/PATTERN.' (1)
'[AXIS/]s/[TICKS/]MAJOR/LENGTH.' (2)

-----------------------------------------------

4.48 "[AXIS/]s/[TICKS/]MAJOR/SPACING.".

Simplest form of name: 's/MA/SP.'
A group of three parameters describing the way in which major tick marks, if any, are to be spaced along the axis specified by "s". Subgroups and the number of parameters in each are as follows:

- \[AXIS/\]s/[TICKS/]MAJOR/[SPACING/]TYPE.\] (1)
- \[AXIS/\]s/[TICKS/]MAJOR/[SPACING/]BASE.\] (1)
- \[AXIS/\]s/[TICKS/]MAJOR/[SPACING/]COUNT.\] (1)

4.49 \[AXIS/\]s/[TICKS/]MAJOR/[SPACING/]TYPE.\]

Simplest form of name: \s/MA/TY.\] A floating-point number specifying where major tick marks are to be placed along the axis specified by "s" (that is to say, at what values in the label coordinate system along that axis). Let "b" represent the value of the parameter \.../BASE.\] (described next) and "k" represent an arbitrary integer. Then, there are six acceptable values of \.../TYPE.\]:

- The value "0." specifies that no major tick marks are to be drawn on the axis.
- The value "1." specifies major tick marks at values of the form plus or minus b times k.
- The value "2." specifies major tick marks at values of the form plus or minus b times 10 to the power k.
- The value "3." specifies major tick marks at values of the form plus or minus b to the power k.
- The value "null 1" specifies that AUTOGRAPH should use a value "1.", "2.", or "3." - whichever it considers best.
- The value "null 2" specifies that AUTOGRAPH should use a value "1.", "2.", or "3." - whichever it considers best - and replace the "null 2" by that value.

Notice that major tick marks on a linear axis may be spaced logarithmically and that major tick marks on a logarithmic axis may be spaced linearly; this is sometimes useful.

Default value: "1.E36" ("null 1") for all "s" (AUTOGRAPH spaces major tick marks as it sees fit).

4.50 \[AXIS/\]s/[TICKS/]MAJOR/[SPACING/]BASE.\] Simplest form of name: \s/MA/BA.\]
A floating-point number which, if greater than zero and non-null, specifies the base value ("b", in the preceding parameter description) used in spacing major tick marks in the label coordinate system along the axis specified by "s". A negative or zero value suppresses major tick marks on the axis. A null value causes AUTOGRAPH to pick an appropriate base value and, if the null was a "null 2", to backstore that value in place of the "null 2".

Default value: "1.E36" ("null 1") for all "s" (AUTOGRAPH picks the base values).

4.51 `[AXIS/]s/[TICKS/]MAJOR/[SPACING/]COUNT.`

Simplest form of name: `s/MA/CO.`

A floating-point number, having an integral value "n" greater than or equal to 0. A negative value is treated as if it were a zero. The value n is only used when major tick marks are to be spaced linearly and the base value ("b", in the preceding parameter descriptions) is to be chosen by AUTOGRAPH. In this case, n is a rough estimate of the minimum number of major tick marks to be placed on the axis specified by "s". The actual number used may vary between "n+2" and "5n/2+4" (approximately).

Default value: "6." for all "s" (somewhere between 8 and 19 major tick marks per linear axis).

4.52 `[AXIS/]s/[TICKS/]MAJOR/PATTERN.`

Simplest form of name: `s/MA/PA.`

A floating-point number specifying the dashed-line pattern to be used for major tick marks on the axis specified by "s". Normally, its integer equivalent is a 16-bit integer in which "0" bits specify "pen-up" segments (gaps) 3 plotter units long and "1" bits specify "pen-down" segments (solids) 3 plotter units long. The value "0." turns off the major tick marks, the value "65535." (decimal) = "177777." (octal) makes them solid lines. If the value "null 1" is used, the next call to AGSTUP resets it to "65535." (decimal).

Default value: "1.E36" ("null 1") for all "s" (solid-line patterns).

4.53 `[AXIS/]s/[TICKS/]MAJOR/LENGTH.`

Simplest form of name: `s/MA/LE.`

A group of two parameters determining the length of the outward-pointing and inward-pointing portions of the major tick marks on the axis specified by "s". Subgroups and the number of parameters in each are as follows:
4.54 `[AXIS]/s/[TICKS]/MAJOR/[LENGTH]/OUTWARD`.

Simplest form of name: `s/MA/OU`.

A floating-point number specifying the length of the outward-pointing portion of each major tick mark on the axis specified by `s`. The value must be of the form `e`, `1.0e`, or `-e`, where `e` is greater than or equal to `0` and less than `1` and represents a fraction of the smaller dimension of the grid window.

Note: "Outward" is defined relative to the normal position of the axis `s`, even when that axis has been moved away from its normal position.

- When a value `e` is used, each major tick mark extends outward `e` units from the axis.
- When a value `1.0e` is used, each major tick mark extends outward to the farther edge of the grid window and then `e` units beyond that edge. (If the axis is not moved away from its normal position, `1.0e` has the same effect as `e`.)
- When a value `-e` is used, the first `e` units of the inward-pointing portion of each major tick mark are erased. (This can be used to create off-axis major tick marks - for whatever that may be worth.)

Default value: `0` for all `s` (all major ticks point inward).

4.55 `[AXIS]/s/[TICKS]/MAJOR/[LENGTH]/INWARD`.

Simplest form of name: `s/MA/IN`.

A floating-point number specifying the length of the inward-pointing portion of each tick mark on the axis specified by `s`. The value must be of the form `e`, `1.0e`, or `-e`, where `e` is greater than or equal to `0` and less than `1` and represents a fraction of the smaller dimension of the grid window.

Note: "Inward" is defined relative to the normal position of the axis `s`, even when that axis has been moved away from its normal position.

- When a value `e` is used, each major tick mark extends inward `e` units from the axis.
- When a value `1.0e` is used, each major tick mark extends inward to the farther edge of the grid window and then `e` units beyond that edge. This feature is used to create grid backgrounds.
When a value "-e" is used, the first "e" units of the outward-pointing portion of each major tick mark are erased.

Default value: ".015" for all "s" (all major ticks point inward).

---

4.56 "[AXIS/]s/[TICKS/]MINOR."

Simplest form of name: "s/MI."

A group of four parameters describing the minor tick marks, if any, which are to be a part of the axis specified by "s". Subgroups and the number of parameters in each are as follows:

\[ \text{Simplest form of name: } 's/MI.' \]

\[ \text{Subgroups and the number of parameters in each are as follows: } \]

\[ 's/[TICKS/]MINOR/SPACING.' (1) \]
\[ 's/[TICKS/]MINOR/PATTERN.' (1) \]
\[ 's/[TICKS/]MINOR/LENGTH.' (2) \]

---

4.57 "[AXIS/]s/[TICKS/]MINOR/SPACING."

Simplest form of name: "s/MI/SP."

A floating-point number specifying the desired number of minor tick marks to be distributed between each pair of major tick marks on the axis specified by "s". Acceptable values are as follows:

* A value less than "1." suppresses minor tick marks completely.
* A value greater than or equal to "1." which is non-null should be integral; it specifies the number of minor tick marks directly.
* The values "null 1" and "null 2" specify that AUTOGRAPH is to choose a reasonable integral value; if a "null 2" is specified, it is replaced by the integral value chosen.

The minor tick marks, if any, are spaced linearly in the label coordinate system along the axis specified by "s". Note that the appropriate value for the usual sort of logarithmic axis is "8."; this causes the minor tick marks between two major tick marks at label-system values \(10^n\) and \(10^{n+1}\) to be placed at the label-system values \(2\times10^n, 3\times10^n, 4\times10^n, \ldots, 9\times10^n\).

Default value: "1.E36" ("null 1") for all "s" (AUTOGRAPH chooses appropriate values).

---

4.58 "[AXIS/]s/[TICKS/]MINOR/PATTERN."
Simplest form of name: 's/MI/PA.'

A floating-point number specifying the dashed-line pattern to be used for minor tick marks on the axis specified by "s"; analogous to "[AXIS/]s/[TICKS/]MAJOR/PATTERN.", described above.

Default value: "1.E36" ("null 1") for all "s" (solid-line patterns).

4.59 `[AXIS/]s/[TICKS/]MINOR/LENGTH.`

Simplest form of name: `s/MI/LE.`

A group of two parameters determining the length of the outward-pointing and inward-pointing portions of the minor tick marks on the axis specified by "s". Subgroups and the number of parameters in each are as follows:

`[AXIS/]s/[TICKS/]MINOR/[LENGTH/]OUTWARD.` (1)
`[AXIS/]s/[TICKS/]MINOR/[LENGTH/]INWARD.` (1)

4.60 `[AXIS/]s/[TICKS/]MINOR/[LENGTH/]OUTWARD.`

Simplest form of name: `s/MI/OU.`

A floating-point number specifying the length of the outward-pointing portion of each minor tick mark on the axis specified by "s"; analogous to "...MAJOR/[LENGTH/]OUTWARD.", described above.

Default value: "0." for all "s" (all minor ticks point inward).

4.61 `[AXIS/]s/[TICKS/]MINOR/[LENGTH/]INWARD.`

Simplest form of name: `s/MI/IN.`

A floating-point number specifying the length of the inward-pointing portion of each minor tick mark on the axis specified by "s"; analogous to "...MAJOR/[LENGTH/]INWARD.", described above.

Default value: ".010" for all "s" (all minor ticks point inward).

4.62 `[AXIS/]s/NUMERIC.`

Simplest form of name: `s/NU.`
A group of eight parameters describing the numeric labels, if any, which are to be a part of the axis specified by "s". Subgroups and the number of parameters in each are as follows:

- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{TYPE.}\.\) (1)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{EXPONENT.}\.\) (1)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{FRACTION.}\.\) (1)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{ANGLE.}\.\) (2)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{OFFSET.}\.\) (1)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{WIDTH.}\.\) (2)

4.63 \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{TYPE.}\.\)

Simplest form of name: ‘s/TYPE.’

The three parameters

- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{TYPE.}\.\)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{EXPONENT.}\.\)
- \([\text{AXIS/}]s/\text{[NUMERIC/]}\text{FRACTION.}\.\)

will be described together, because they are so closely interdependent. They specify the type of numeric labels to be used (at major-tick positions) on the axis specified by "s". A fourth parameter,

\([\text{AXIS/}]s/\text{[TICKS/]}\text{MAJOR/SPACING.} \text{TYPE.}\.\)

described above, also affects the type of numeric labels to be used. I shall refer to these four parameters in the ensuing discussion using short forms of their names (‘s/TYPE.’, ‘s/EXPO.’, ‘s/FRAC.’, and ‘s/MAJOR/TYPE.’, respectively).

All four have the default value "null 1" (except for the first, which has the default value "0." for "s" = "RIGHT" and "TOP"), leaving AUTOGRAPH free to choose values which are consistent with each other and with other parameters describing the axis specified by "s". Any one or more of them may be given the value "null 2" (in which case an actual value chosen by AUTOGRAPH is back-stored over the "null 2") or an actual integral floating-point value. Acceptable actual values are as follows:

- Setting ‘s/TYPE.’ to "0." turns off the numeric labels on the axis specified by "s". The other three parameters are then ignored.

- Setting ‘s/TYPE.’ to "1." selects "scientific" notation. Each numeric label is written in the form

\([-] [i] [. ] [f] \times 10^e\)

where brackets enclose portions which may be independently present or absent and "e" is a superscript exponent.
The parameter `.s/EXPO.` specifies the length of "i" (the number of characters), thus also specifying the value of the exponent "e". If `.s/EXPO.` has a value less than or equal to zero, "i" is omitted. If `.s/EXPO.` is less than zero and has the integral absolute value "n", the fraction "f" is forced to have "n" leading zeroes.

The parameter `.s/FRAC.` specifies the length of "f" (the number of characters). If `.s/FRAC.` is less than or equal to zero, "f" is omitted. If `.s/FRAC.` is less than zero, the decimal point is omitted.

If "[i] [.1 [f]" has the value "1.", the first part of the label is omitted, leaving only "10 e".

If the entire label has the value "0.", the single character "0" is used.

The value of `.s/MAJOR/TYPEx.` is immaterial.

* Setting `.s/TYPEx.` to "2." selects "exponential" notation, the exact nature of which depends on the value of `.s/MAJOR/TYPEx.` as follows:

  * If `.s/MAJOR/TYPEx.` has the value "1." (all major ticks at values of the form plus or minus b times k), each numeric label is written in the form

    \[-\] [i] [.][f] x 10 e

    where brackets enclose portions which may be independently present or absent and "e" is a superscript exponent.

    The parameter `.s/EXPO.` specifies the integral value of the exponent "e".

    The parameter `.s/FRAC.` specifies the length of "f" (the number of characters). If `.s/FRAC.` is less than or equal to zero, f is omitted. If `.s/FRAC.` is less than zero, the decimal point is omitted.

    If the label value is exactly zero, the single character "0" is used.

  * If `.s/MAJOR/TYPEx.` has the value "2." (all major ticks at values of the form plus or minus b times 10 to the power k), each numeric label is written in the form

    \[-\] [i] [.][f] x 10 e

    where brackets enclose portions which may be independently present or absent and "e" is a superscript exponent.

    The parameter `.s/EXPO.` specifies the integral value of the exponent "e" when "k" equals "0." The value of "e" is `.s/EXPO.` plus "k".

    The parameter `.s/FRAC.` specifies the length of "f" (the number of characters). If `.s/FRAC.` is less than or equal to zero, "f" is
omitted. If s/FRAC. is less than zero, the decimal point is omitted.

If the label value is exactly zero, the single character "0" is used.

If s/MAJOR/TYP. has the value "3." (all major ticks at values of the form plus or minus b to the power k), each numeric label is written in the form

\[-[i][.][f]e\]

where brackets enclose portions which may be independently present or absent and "e" is a superscript exponent.

The parameter s/EXPO. is ignored. The value of "e" is "k".

The parameter s/FRAC. specifies the length of "f" (the number of characters). If s/FRAC. is less than or equal to zero, "f" is omitted. If s/FRAC. is less than zero, the decimal point is omitted.

Note that "[i][.][f]" expresses the value of "b".

Setting s/TYP. to "3." selects "no-exponent" notation, the exact nature of which depends on the value of s/MAJOR/TYP., as follows:

If s/MAJOR/TYP. has the value "1." (all major ticks at values of the form plus or minus b times k), each numeric label is written in the form

\[-[i][.][f]\]

where brackets enclose portions which may be independently present or absent.

The parameter s/EXPO. is ignored.

The parameter s/FRAC. specifies the length of "f" (the number of characters). If s/FRAC. is less than or equal to zero, "f" is omitted. If s/FRAC. is less than zero, the decimal point is omitted.

If the label value is exactly zero, the single character "0" is used.

If s/MAJOR/TYP. has the value "2." (all major ticks at values of the form plus or minus b times 10 to the power k), each numeric label is written in the form

\[-[i][.][f]\]

where brackets enclose portions which may be independently present
The parameter ’s/EXPO.’ is ignored.

The length of "f" (the number of characters) is specified by the function

\[
\text{MAX}(\text{’s/FRAC.’},0) - k
\]

if this quantity is greater than zero, and

\[
\text{MIN}(\text{’s/FRAC.’},0)
\]

otherwise. This may appear somewhat formidable, but it produces a simple, desirable result. Suppose, for example, that ’s/FRAC.’ = "1.", "b" = "3.6", and "k" ranges from "-3" to "+3"; the labels produced are

.0036, .036, .36, 3.6, 36., 360., and 3600.

The parameter ’s/FRAC.’ may be viewed as specifying the length of "f" when "k" is zero. If the function value is less than or equal to zero, "f" is omitted; if it is less than zero, the decimal point is omitted.

• If ’s/MAJOR/TYP.’ has the value "3." (all major ticks at values of the form plus or minus b to the power k), each numeric label is written in the form

\[-] [i] [.] [f]

if "k" is greater than or equal to zero, and in the form

\[-] l/ [i] [.] [f]

if "k" is less than zero. Brackets enclose portions which may be independently present or absent.

The parameter ’s/EXPO.’ is ignored.

The length of "f" (the number of characters) is specified by the function

\[
\text{’s/FRAC.’} \times \text{ABS}(k)
\]

if "k" is non-zero, or

\[
\text{MIN}(\text{’s/FRAC.’},0)
\]

if "k" is zero. Again, this function produces a simple result. Suppose that ’s/FRAC.’ = "1.", "b" = "1.1", and "k" ranges from "-3" to "+3"; the labels produced are
1/1.331, 1/1.21, 1/1.1, 1., 1.1, 1.21, and 1.331

The parameter `s/FRAC.` may be viewed as specifying the length of 
"f" when "k" is equal to 1. If the function value is less than or 
equal to zero, "f" is omitted; if it is less than zero, the decimal 
point is omitted.

Another example: Suppose `s/FRAC. = "-1.", "b" = "2.", and "k" 
ranges from "-4" to "+4"; the labels produced are 

1/16, 1/8, 1/4, 1/2, 1, 2, 4, 8, and 16

Default value: "1.E36" ("null 1") for all three for all "s" (AUTOGRAPH 
chooses values to use), except for

`RIGHT/[NUMERIC/TYPE.` and
`TOP/[NUMERIC/TYPE.`

which are zeroed to suppress the numeric labels on the right and top axes.

4.64 `[AXIS/ls/[NUMERIC/EXPONENT.`.

Simplest form of name: `s/EX.`

See the discussion of `[AXIS/ls/[NUMERIC/TYPE.` above.

4.65 `[AXIS/ls/[NUMERIC/FRACtion.`.

Simplest form of name: `s/FR.`

See the discussion of `[AXIS/ls/[NUMERIC/TYPE.` above.

4.66 `[AXIS/ls/[NUMERIC/ANGLE.`.

Simplest form of name: `s/AN.`

A group of two integral floating-point numbers specifying the orientation 
angle of numeric labels on the axis specified by "s". Subgroups and the 
number of parameters in each are as follows:

`[AXIS/ls/[NUMERIC/ANGLE/LST.` (1)
`[AXIS/ls/[NUMERIC/ANGLE/2ND.` (1)
4.67 \([\text{AXIS/s}]\text{/[NUMERIC/]ANGLE/1ST.}\)\.

Simplest form of name: \(\text{s/AN/1S.}\)

An integral floating-point number having one of the values "0.", "90.", "180.", or "270." (plus or minus a small multiple of "360.") specifying the user's first choice for the orientation angle of numeric labels on the axis specified by "s". The value is stated in degrees counter-clockwise from a left-to-right horizontal vector.

The routine AGSTUP decides whether the first choice or the second choice is to be used. The second choice is used only when the first choice leads to overlap problems and the current value of \([\text{AXIS/s}]\text{/CONTROL.}\) is a "3." or a "4." and the second choice works out better than the first. If AGSTUP decides to use the first choice, it leaves the first-choice parameter with a positive value; if it decides to use the second choice, it leaves the first-choice parameter with a negative value. Values are made positive or negative by adding and subtracting multiples of "360."

Default value: "0." for all "s" (horizontal labels preferred on all axes).

4.68 \([\text{AXIS/s}]\text{/[NUMERIC/]ANGLE/2ND.}\)\.

Simplest form of name: \(\text{s/AN/2N.}\)

An integral floating-point number having one of the values "0.", "90.", "180.", or "270." (plus or minus a small multiple of "360.") specifying the user's second choice for the orientation angle of numeric labels on the axis specified by "s". The value is stated in degrees counter-clockwise from a left-to-right horizontal vector. See the description of the preceding parameter.

Default value: "90." for all "s" (vertical labels, readable from the right, on all axes).

4.69 \([\text{AXIS/s}]\text{/[NUMERIC/]OFFSET.}\)\.

Simplest form of name: \(\text{s/OF.}\)

A floating-point number specifying the desired position of numeric labels relative to the axis specified by "s".

If the value is positive, numeric labels are to be placed toward the outside of the grid. If the value is negative, numeric labels are to be placed toward the inside of the grid. In either of these two cases, the magnitude of the value specifies the distance from the line portion of the axis to the nearest part of any numeric label, stated as a fraction of the smaller dimension of the grid window. Note: "Inside" and "outside" are defined relative to the normal position of the axis "s", even when that axis has been moved away from
its normal position.

If the value is exactly zero, each numeric label is centered on the axis. In this case, the line portion of the axis is suppressed and major and minor tick marks are moved outward so as not to overlap the numeric labels.

Default value: ".015" for all "s" (all labels outside the grid).

4.70 \[AXIS/\]s/[NUMERIC/]/WIDTH.

Simplest form of name: \(s/WI\).

A group of two floating-point parameters specifying the widths of characters to be used in numeric labels on the axis specified by "s". Subgroups and the number of parameters in each are as follows:

\[AXIS/\]s/[NUMERIC/]/WIDTH/MANTISSA. (1)
\[AXIS/\]s/[NUMERIC/]/WIDTH/EXponent. (1)

4.71 \[AXIS/\]s/[NUMERIC/]/WIDTH/MANTISSA.

Simplest form of name: \(s/WM\).

A floating-point number specifying the width of characters to be used in the mantissa of each numeric label on the axis specified by "s", expressed as a fraction of the smaller dimension of the grid window.

Default value: ".015" for all "s".

4.72 \[AXIS/\]s/[NUMERIC/]/WIDTH/EXponent.

Simplest form of name: \(s/WI/EX\).

A floating-point number specifying the width of characters to be used in the exponent of each numeric label on the axis specified by "s", expressed as a fraction of the smaller dimension of the grid window.

Default value: ".010" for all "s".

4.73 \'DASH\'.

Simplest form of name: \'DAS\'.

PARAMETERS - 74 -
A group of thirty parameters, the first of which determines what dashed-line patterns are to be used by the routines EZMY and EZMXY and the rest of which describe the "user" set of dashed-line patterns (as opposed to the "alphabetic" set, which is defined by code in the subroutine AGCURV and is not subject to change by the user). Subgroups and the number of parameters in each are as follows:

- `DASH SELECTOR.` (1)
- `DASH LENGTH.` (1)
- `DASH CHARACTER.` (1)
- `DASH DOLLAR-QUOTE.` (1)
- `DASH PATTERNS.` (26)

4.74 `DASH SELECTOR.`

Simplest form of name: `DAS SE`.

The parameter `DASH SELECTOR.` is given a negative integral value to specify that the routines EZMY and EZMXY should use the "alphabetic" set of 26 dashed-line patterns for the curves they draw and a positive integral value "n", less than or equal to 26, to specify that EZMY and EZMXY should use the first "n" patterns in the "user" set of dashed-line patterns, as defined by the current values of the remaining parameters in the group `DASH`.

Each of the patterns in the "alphabetic" set specifies a solid line interrupted periodically by a letter of the alphabet. Each of the patterns in the "user" set is as defined by the user. The default "user" set produces all solid lines.

The routines EZY and EZXY, which draw but one curve per call, always use the first of the patterns in the "user" set; they are unaffected by the value of `DASH SELECTOR.`.

The selected pattern set is used in a circular fashion. For example, if `DASH SELECTOR.` has the value "3." and EZMY is used to draw nine curves, pattern 1 is used for curves 1, 4, and 7, pattern 2 for curves 2, 5, and 8, and pattern 3 for curves 3, 6, and 9.

Default value: "+1." (The first element of the "user" set of dashed-line patterns is to be used by EZMY and EZMXY.)

4.75 `DASH LENGTH.`

Simplest form of name: `DAS LE`.

An integral floating-point number specifying how long character-string dashed-line patterns are expected to be. In a user call to ANOTAT with a positive fifth argument (implying that the sixth argument is an array of character-
string dashed-line patterns) or in a user call to AGSEMT setting 
"DASH/PATTERN/n." (in which case the second argument is such a pattern), the 
specified character strings must be of the length specified by the current 
value of "DASH/LENGTH.".

Default value: "8." (dashed-line patterns are expected to be eight characters 
long).

4.76 "DASH/CHARACTER.".

Simplest form of name: "DAS/CH."

A floating-point number specifying the width of each character (other than a 
dollar sign or a quote) which is drawn along a curve as directed by a 
character-string dashed-line pattern (whether from the "alphabetic" set or 
from the "user" set). This width is expressed as a fraction of the smaller 
dimension of the grid window.

Default value: ".010"

4.77 "DASH/DOLLAR-QUOTE.".

Simplest form of name: "DAS/DO."

A floating-point number specifying the line length corresponding to a dollar 
sign (solid) or a quote (gap) in a character-string dashed-line pattern, 
expressed as a fraction of the smaller dimension of the grid window.

Default value: ".010"

4.78 "DASH/PATTERNS.".

Simplest form of name: "DAS/PA."

A group of 26 parameters defining the "user" set of dashed-line patterns. 
Subgroups and the number of parameters in each are as follows:

'\n\nDASH/PATTERNS/1.' (1) 
'DASH/PATTERNS/2.' (1) 
\n\n'DASH/PATTERNS/26.' (1)
4.79 'DASH/PATTERNS/n'.

Simplest form of name: 'DAS/PA/n.'

(The symbol "n" represents an integer between "1" and "26", inclusive.) An integral floating-point number defining the "n"th dashed-line pattern in the "user" set.

If the value is positive, it must be between "0." and "65535."., inclusive, and is interpreted as a 16-bit binary pattern in which each "0" bit specifies a "pen-up" gap segment 3 plotter units long and each "1" bit specifies a "pen-down" solid segment 3 plotter units long. Such a pattern may be defined by a user call to AGSETI or AGSETF.

If the value is negative, it serves as an identifier, allowing AUTOGRAPH to retrieve, from its character storage space, a character string in which each single quote specifies a "pen-up" gap segment, each dollar sign specifies a "pen-down" solid segment, and each other character is simply to be drawn as a part of the line. Such a pattern may be defined by a user call to AGSETC.

Note that the function "AGDSHN" allows a user to easily generate the name of the "n"th dash pattern.

Default values: "65535." for all "n" (solid lines).

4.80 'LABEL.'.

Simplest form of name: 'LAB.'

A group of 3+10*n parameters, where "n" is the current value of 'LABEL/BUFFER/LENGTH.' (8, by default) describing up to "n" informational labels. These labels are a part of the background drawn by a call to the routine AGBACK. Subgroups and the number of parameters in each are as follows:

'LABEL/CONTROL.' (1)
'LABEL/BUFFER/LENGTH.' (1)
'LABEL/BUFFER/CONTENTS.' (10*n)
'LABEL/NAME.' (1)

4.81 'LABEL/CONTROL.'.

Simplest form of name: 'LAB/CO.'

An integral floating-point number having the value "0.", "1.", or "2.". Values greater than "2." are changed to a "2." by the next AGSTUP call. Values less than "0." are changed to a "0." by the next AGSTUP call; negative values have a special use, however (see below).
The value "0." disables the drawing of informational labels. They remain defined, however.

The value "1." enables the drawing of informational labels and specifies that they may not be shrunk in response to overlap problems.

The value "2." enables the drawing of informational labels and specifies that they may be shrunk in response to overlap problems.

Default value: "2." (labels enabled, shrinkable).

Special action by AGSETP: An AGSETP call which sets this parameter (individually, rather than as part of a group) to a negative value results in the deletion of all currently-defined labels. Note that the negative value is changed to a zero by the next AGSTUP call; thus, the drawing of informational labels is disabled until re-enabled by the user.

4.82 `LABEL/BUFFER.`.

Simplest form of name: `LAB/BU.`

A group of 1+10*n parameters, where "n" is the current value of `LABEL/BUFFER/LENGTH.` (8, by default). Subgroups and the number of parameters in each are as follows:

`LABEL/BUFFER/LENGTH.` (1)
`LABEL/BUFFER/CONTENTS.` (10*n)

4.83 `LABEL/BUFFER/LENGTH.`.

Simplest form of name: `LAB/BU/LE.`

An integral floating-point number specifying the number of 10-word label definitions the label buffer will hold. A user program may need to retrieve, but must not set, the value of this parameter, since its value must match the second dimension of the label buffer.

Increasing the size of the label buffer requires modifying the AUTOGRAPH source code. See the paragraph "INFORMATIONAL LABELS", in the section "OVERVIEW".

Default value: "8."

4.84 `LABEL/BUFFER/CONTENTS.`.
Simplest form of name: 'LAB/BU/CO.'

This parameter group may be thought of as an array FLLB, dimensioned 10xn, containing up to n 10-word label definitions. For a second subscript j,

- FLLB(1,j) is either a floating-point "0.", saying that no label is defined by this 10-word block, or it is non-zero, in which case it identifies a character string in AUTOGRAPH's character-string storage area; the character string serves as a name for the label defined by this 10-word block. When FLLB(1,j) is non-zero:
  - FLLB(2,j) is either a "0.", to enable drawing of the label, or a "1.", to disable drawing of the label,
  - FLLB(3,j) and FLLB(4,j) are the x and y coordinates of the label's "basepoint", in the grid coordinate system,
  - FLLB(5,j) and FLLB(6,j) are the x and y components of the label's "offset vector", stated as signed fractions of the smaller dimension of the grid window,
  - FLLB(7,j) is an integral floating-point number "0.", "90.", "180.", or "270.", specifying the angle at which the label's "baseline" emanates from the end of its offset vector,
  - FLLB(8,j) is an integral floating-point number specifying how the lines of the label are to be positioned relative to the end of the offset vector ("-1." to line up the left ends, "0." to line up the centers, or "+1." to line up the right ends),
  - FLLB(9,j) is an integral floating-point count of the number of lines belonging to the label, and
  - FLLB(10,j) is an integral floating-point pointer specifying the second subscript (in the line buffer) of the first line of the label (the one having the largest line number), or, if no lines belong to the label, a "0.".

It is not recommended that a user program change the contents of this buffer directly. Label definitions should be accessed indirectly by means of the parameters 'LABEL/NAME.' and 'LABEL/[DEFINITION/]...'.

Default values: The label buffer contains four pre-defined labels, corresponding to the four edges of the grid window. They are as follows:
The description of 'LINE/BUFFER/CONTENTS.', below, gives the default values for the definitions of the lines which belong to these labels.

---------------------------------------------

4.85 'LABEL/BUFFER/NAMES.'.

Simplest form of name: 'LAB/BU/NA.'

This group is a subset of the previous one. It provides a way of retrieving the names of all currently-defined labels.

---------------------------------------------

4.86 'LABEL/NAME.'.

Simplest form of name: 'LAB/NA.'

An integral floating-point pointer which, if non-zero, specifies a particular label in the label buffer - the one which is to be referenced by the parameter group 'LABEL/DEFINITION.' (which see, below).

Setting 'LABEL/NAME.' is the required first step in accessing a particular label definition.

Default value: "0." (undefined).

Special action by AGSETP: To access the definition of a particular label, one must first call AGSETP with 'LABEL/NAME.' as the first argument and the name of the label one wishes to access as the second argument. This causes AGSETP (which is called by AGSETC) to search for the definition of the desired label in the label buffer. If that definition is not found, a new one is made up and inserted in the label buffer. In either case, 'LABEL/NAME.' is given a floating-point value whose integer equivalent specifies the second subscript of the label definition in the label buffer.

The definition of a new label has the name specified by the user, a suppression flag "0.", a basepoint (.5,.5), an offset vector (0.,0.), a baseline angle "0.", a centering option "C.", a line count "0.", and a first-line index "0.".

-------------------------------------------------------------------

PARAMETERS  -  80  -  AUTOGRAPH
4.87 'LABEL/DEFINITION.'.

Simplest form of name: 'LAB/DE.'

A set of nine parameters defining the label specified by the current value of 'LABEL/NAME.' If 'LABEL/NAME.' has the value "0.", referencing this group or a parameter in it causes an error exit. Subgroups and the number of parameters in each are as follows:

'LABEL/[DEFINITION/]SUPPRESSION.' (1)
'LABEL/[DEFINITION/]BASEPOINT.' (2)
'LABEL/[DEFINITION/]OFFSET.' (2)
'LABEL/[DEFINITION/]ANGLE.' (1)
'LABEL/[DEFINITION/]CENTERING.' (1)
'LABEL/[DEFINITION/]LINES.' (1)
'LABEL/[DEFINITION/]INDEX.' (1)

4.88 'LABEL/[DEFINITION/]SUPPRESSION.'.

Simplest form of name: 'LAB/SU.'

An integral floating-point "suppression flag" having the value "0." or "1." and specifying whether drawing of the label specified by 'LABEL/NAME.' is enabled ("0.") or disabled ("1.").

Default value for a new label: "0." (label enabled).

Special action by AGSETP: If a user program attempts to set this parameter (individually, rather than as part of a group) to a negative value, the lines of the label specified by 'LABEL/NAME.' are deleted and 'LINE/NUMBER.' is zeroed. If the negative value is less than "-1.", the label is deleted as well and "LABEL/NAME." is zeroed. (Deleting a label means that its name cell is set to "0.".)

4.89 'LABEL/[DEFINITION/]BASEPOINT.'.

Simplest form of name: 'LAB/BA.'

A set of two parameters specifying the x and y coordinates of the basepoint of the label specified by 'LABEL/NAME.', in the grid coordinate system. The label is positioned relative to this basepoint. Subgroups and the number of parameters in each are as follows:

'LABEL/[DEFINITION/]BASEPOINT/X.' (1)
'LABEL/[DEFINITION/]BASEPOINT/Y.' (1)
4.90 "LABEL/[DEFINITION/]BASEPOINT/X.".

Simplest form of name: ‘LAB/BA/X.’

The x coordinate of the basepoint of the label specified by ‘LABEL/NAME.’.
The value "0." refers to the left edge of the grid window, the value "1." to the right edge of the grid window.

Default value for a new label: ".5" (centered).

4.91 "LABEL/[DEFINITION/]BASEPOINT/Y.".

Simplest form of name: ‘LAB/BA/Y.’

The y coordinate of the basepoint of the label specified by ‘LABEL/NAME.’.
The value "0." refers to the bottom edge of the grid window, the value "1." to the top edge of the grid window.

Default value for a new label: ".5" (centered).

4.92 "LABEL/[DEFINITION/]OFFSET.".

Simplest form of name: ‘LAB/OF.’

A set of two parameters specifying the x and y components of the offset vector of the label specified by ‘LABEL/NAME.’, as signed fractions of the smaller dimension of the grid window. The offset vector has its basepoint at the label basepoint. Subgroups and the number of parameters in each are as follows:

‘LABEL/[DEFINITION/]OFFSET/X.’ (1)
‘LABEL/[DEFINITION/]OFFSET/Y.’ (1)

4.93 "LABEL/[DEFINITION/]OFFSET/X.".

Simplest form of name: ‘LAB/OF/X.’

The x component of the offset vector of the label specified by ‘LABEL/NAME.’ - negative toward the left edge, positive toward the right edge, of the grid window. The magnitude represents a fraction of the smaller dimension of the grid window.

Default value for a new label: "0." (zero-length vector).
4.94 "LABEL/[DEFINITION/]OFFSET/Y.".

Simplest form of name: "LAB/OF/Y."

The y component of the offset vector of the label specified by "LABEL/NAME." — negative toward the bottom edge, positive toward the top edge, of the grid window. The magnitude represents a fraction of the smaller dimension of the grid window.

Default value for a new label: "0." (zero-length vector).

4.95 "LABEL/[DEFINITION/]ANGLE."

Simplest form of name: "LAB/AN."

An integral floating-point number having one of the values "0.", "90.", "180.", or "270.", and specifying the direction in which the baseline of the label specified by "LABEL/NAME." emanates from the end of its offset vector, measured counter-clockwise from a left-to-right horizontal vector. All the lines of a label are written parallel to its baseline and in the direction of the baseline.

Default value for a new label: "0." (horizontal, left to right).

4.96 "LABEL/[DEFINITION/]CENTERING."

Simplest form of name: "LAB/CE."

An integral floating-point number specifying the alignment of the lines of the label specified by "LABEL/NAME." with the end of its offset vector. A negative value aligns the left ends, a zero value the centers, and a positive value the right ends, of the lines.

Default value for a new label: "0." (centers aligned).

4.97 "LABEL/[DEFINITION/]LINES."

Simplest form of name: "LAB/LI."

An integral floating-point number specifying the number of lines in the label specified by "LABEL/NAME.".

This parameter is updated by AUTOGRAPH as lines are added to or deleted from the label and should not be set by a user program.

Default value for a new label: "0." (no lines).
4.98 `LABEL/[DEFINITION/]INDEX`.

Simplest form of name: `LAB/IN`.

An integral floating-point number specifying the second subscript (in the line buffer) of the first line belonging to the label specified by `LABEL/NAME` - a zero if no line belongs to the label.

This parameter is updated by AUTOGRAPH as lines are added to or deleted from the label and should not be set by a user program.

Default value for a new label: "0." (no lines).

4.99 `LINE`.

Simplest form of name: `LIN`.

A group of 4+6*n parameters, where "n" is the current value of `LINE/BUFFER/LENGTH` (16, by default) describing up to "n" lines, each of which is a part of some informational label. Subgroups and the number of parameters in each are as follows:

- `LINE/MAXIMUM` (1)
- `LINE/END` (1)
- `LINE/BUFFER/LENGTH` (1)
- `LINE/BUFFER/CONTENTS` (6*n)
- `LINE/NUMBER` (1)

4.100 `LINE/MAXIMUM`.

Simplest form of name: `LIN/MA`.

An integral floating-point number specifying the assumed maximum length of a character string delivered to AUTOGRAPH for use as the text of a label line. Such a character string may occur as the first argument of a call to ANOTAT (defining the text of line "100." in the label "L"), as the second argument of a call to ANOTAT (defining the text of line "-100." in the label "B"), as the last argument of a call to one of the routines EZY, EZXY, EZMY, or EZMXY (defining the text of line "100." in the label "T"), or as the second argument of a call to AGSETC whose first argument is `LINE/[DEFINITION/]TEXT`.

(defining the text of any line). In each of these cases, the character string must be of the length specified by `LINE/MAXIMUM` or shorter. If it is shorter, its last character must be the character specified by `LINE/END`.

PARAMETERS - 84 - AUTOGRAPH
described below.

This parameter may be given any desired non-negative integral value.

Default value: "40."

4.101 `LINE/END`.

Simplest form of name: `LIN/EN`.

A character string whose first character is the one used to mark the end of a character string defining the text of a label line (in calls to ANOTAT, EZY, EZXY, EZMY, EZMXY, and AGSETC), when that character string is shorter than the current maximum specified by `LINE/MAXIMUM` (as described above).

The terminator character does not become a part of the text of the line. It is stripped off, so that only the preceding characters constitute the text of the line.

Default value: "$".

4.102 `LINE/BUFFER`.

Simplest form of name: `LIN/BU`.

A group of 1+6*n parameters, where "n" is the current value of `LINE/BUFFER/LENGTH` (16, by default). Subgroups and the number of parameters in each are as follows:

- `LINE/BUFFER/LENGTH` (1)
- `LINE/BUFFER/CONTENTS` (6*n)

4.103 `LINE/BUFFER/LENGTH`.

Simplest form of name: `LIN/BU/LE`.

An integral floating-point number specifying the number of 6-word line definitions the line buffer will hold. A user program may need to retrieve, but must not set, the value of this parameter, since its value must match the second dimension of the line buffer.

Increasing the size of the line buffer requires modifying the AUTOGRAPH source code. See the paragraph "INFORMATIONAL LABELS", in the section "OVERVIEW".

Default value: "16".
Simplest form of name: 'LIN/BU/CO.'

This group may be thought of as an array FLLN, dimensioned 6xn, containing up to n 6-word line definitions. For a second subscript j,

• FLLN(1,j) is either a floating-point "null 1", saying that no label line is defined by this 6-word block, or an integral floating-point "line number", saying that it does define a label line, in which case:
  - FLLN(2,j) is either "0.", to enable drawing of the line, or "1.", to disable drawing of the line.
  - FLLN(3,j) is the floating-point width of each character of the line, stated as a fraction of the smaller dimension of the grid window.
  - FLLN(4,j) is an integral floating-point number serving as the identifier of the character string defining the text of the line and enabling it to be retrieved from AUTOGRAPH's internal character storage space.
  - FLLN(5,j) is an integral floating-point count of the number of characters in the text of the line.
  - FLLN(6,j) is an integral floating-point number specifying the second subscript (in the line buffer) of the next line of the label to which this line belongs (that one of the remaining lines in the chain with the largest line number) or, if there is no next line, a "0.".

It is not recommended that a user program change the contents of this buffer directly. Line definitions should be accessed indirectly by means of the parameters 'LINE/NÚMBER.' and 'LINE/[DEFINITION/].'

Default values: The line buffer contains four pre-defined lines, each of which belongs to one of the four pre-defined labels. They are as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>'L'</th>
<th>'R'</th>
<th>'B'</th>
<th>'T'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line number:</td>
<td>+100.0</td>
<td>-100.0</td>
<td>-100.0</td>
<td>+100.0</td>
</tr>
<tr>
<td>Suppression flag:</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Character width:</td>
<td>0.015</td>
<td>0.015</td>
<td>0.015</td>
<td>0.020</td>
</tr>
<tr>
<td>Text pointed to:</td>
<td>'Y'</td>
<td>'X'</td>
<td>'X'</td>
<td>'X'</td>
</tr>
<tr>
<td>Text length:</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Next-line index:</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The description of 'LABEL/BUFFER/CONTENTS.', above, gives default values for the definitions of the four labels which contain these lines.
4.105 'LINE/NUMBER'.

Simplest form of name: 'LIN/N.'

An integral floating-point pointer which, if non-zero, specifies a particular line in the line buffer - the one which is to be referenced by the parameter group 'LINE/DEFINITION.' (which see, below).

Setting this parameter is the required first step in accessing a particular line definition.

Default value: "0." (undefined).

Special action by AGSETP: To access the definition of a particular line of a particular label, one must ensure that 'LABEL/NAME.' (which see, above) is set. Then, one must call AGSETI with 'LINE/NUMBER.' as the first argument and the number of the line one wishes to access as the second argument. This causes AGSETP (which is called by AGSETI) to search the line buffer for the definition of a line belonging to the label specified by the current value of 'LABEL/NAME.' and having the desired line number. If no such definition is found, a new one is made up, inserted in the line buffer, and linked into the proper place in the chain of lines belonging to the label. In either case, 'LINE/NUMBER.' is given an integral floating-point value specifying the second subscript of the line definition in the line buffer.

The definition of a new line has the number specified by the user, a suppression flag "0.", a character width "0.015", a pointer to the text string "I", and a text length "1.".

Note: The "line numbers" are used to identify the lines of a label and to specify their positions relative to each other and to the baseline of the label. Lines having positive line numbers are drawn above the label baseline, lines having zero line numbers are drawn along the label baseline, and lines having negative line numbers are drawn below the label baseline. A line having a greater line number than another line is drawn above that line.

("Above" and "below" are used here from the viewpoint of someone reading the label.) The magnitudes of the line numbers in no way affect inter-line spacing, which is determined by AUTOGRAPH itself.

4.106 'LINE/DEFINITION'.

Simplest form of name: 'LIN/DE.'

A group of five parameters defining the line specified by 'LINE/NUMBER'. If 'LINE/NUMBER.' has the value "0.", referencing a parameter in this group causes an error exit. Subgroups and the number of parameters in each are as follows:
4.107 "LINE/[DEFINITION/] SUPPRESSION."

Simplest form of name: ˜LIN/SU.˜

An integral floating-point number having the value "0." or "1." and specifying whether drawing of the line specified by "LINE/NUMBER." is enabled ("0.") or disabled ("1.").

Default value for a new line: "0." (line enabled).

Special action by AGSETP: If a user program attempts to set this parameter (individually, rather than as a part of a group) to a negative value, the line specified by "LINE/NUMBER." is deleted and "LINE/NUMBER." is reset to "0." (Deleting a line means that it is unlinked from the chain of lines belonging to its label and that its number cell is set to "null l").

4.108 "LINE/[DEFINITION/] CHARACTER."

Simplest form of name: ˜LIN/CH.˜

A floating-point number specifying the desired width of each character of the line specified by "LINE/NUMBER."; stated as a fraction of the smaller dimension of the grid window.

Default value for a new line: ".015".

4.109 "LINE/[DEFINITION/] TEXT."

Simplest form of name: ˜LIN/TE.˜

An integral floating-point number serving as an identifier for a character string stored away in AUTOGRAPH's internal character storage space.

Default value for a new line: "" "".

Special action by AGSETP: When this parameter is set by a call to AGSETC, the character string appearing as the second argument of AGSETC is stored in a character storage array inside AUTOGRAPH, an identifier allowing for later retrieval of the string is generated, and the value of that identifier is stored (by AGSETP, which is called by AGSETC) as the parameter value. At that
time, the length of the string is determined and 'LINE/[DEFINITION/LENGTH].'
is set. See 'LINE/MAXIMUM.' and 'LINE/TERMINATOR.', above.

4.110 'LINE/[DEFINITION/LENGTH].'

Simplest form of name: 'LIN/LE.'

An integral floating-point count of the number of characters in the text of
the line specified by 'LINE/NUMBER.' Setting this parameter less than or
equal to zero suppresses the drawing of the line. See also the description of
'LINE/[DEFINITION/TEXT]', above.

Default value for a new line: "1." (one character - a blank).

4.111 'LINE/[DEFINITION/INDEX].'

Simplest form of name: 'LIN/IN.'

An integral floating-point number specifying the second subscript (in the line
buffer) of the next line of the label - a zero if there is no next line.

This parameter is updated by AUTOGRAPH as lines are added to or deleted from
the label and should not be set by a user program.

4.112 'SECONDARY.'

Simplest form of name: 'SEC.'

A group of 149 "secondary" control parameters. These are not normally set by
a user program, but are computed by AUTOGRAPH itself (the routine AGSTUP).
Their values may be of use in some applications. Subgroups and the number of
parameters in each are as follows:

'SECONDARY/GRAPH.' (4)
'SECONDARY/USER.' (4)
'SECONDARY/CURVE.' (4)
'SECONDARY/DIMENSIONS.' (3)
'SECONDARY/AXIS.' (80)
'SECONDARY/LABEL.' (54)

4.113 'SECONDARY/GRAPH.'

Simplest form of name: 'SEC/GR.'
A group of four floating-point numbers specifying the x coordinates of the left and right edges of the graph window and the y coordinates of the bottom and top edges of the graph window, in the grid coordinate system. These values are used by AUTOGRAPH to determine whether a point whose coordinates are expressed in the grid coordinate system lies inside or outside the graph window.

If the parameters in the group 'GRID.' have their default values (".15", ".95", ".15", ".95", and "0."), these four parameters will be given the values ".1875", "1.0625", ".1875", and "1.0625". Note that ".1875 = (0-.15)/(.95-.15) and that 1.0625 = (1-.15)/(.95-.15).

4.114 "SECONDARY/USER."

Simplest form of name: 'SEC/US.'

A set of four floating-point numbers specifying the x coordinates of the left and right edges of the grid window and the y coordinates of the bottom and top edges of the grid window, in the user coordinate system. These values are used in mapping user curve points into the grid window. The routines AGSTUP, AGBACK, and AGCURV use these four numbers as arguments 5 through 8 in calls to the system-plot-package routine SET.

4.115 "SECONDARY/CURVE."

Simplest form of name: 'SEC/CU.'

A group of four floating-point numbers specifying the x coordinates of the left and right edges of the grid (curve) window and the y coordinates of the bottom and top edges of the grid (curve) window. The x coordinates are stated as fractions of the distance from left to right, and the y coordinates as fractions of the distance from bottom to top, in the plotter frame. The routines AGSTUP, AGBACK, and AGCURV use these four numbers as arguments 1 through 4 in calls to the system-plot-package routine SET. If the parameters in the groups 'GRAPH.' and 'GRID.' have their default values, these four parameters are given the values ".15", ".95", ".15", and ".95".

4.116 "SECONDARY/DIMENSIONS."

Simplest form of name: 'SEC/DI.'

A group of three floating-point numbers, the first two of which specify the width and height of the grid window and the third of which is equal to the smaller of the first two. Each is stated as a number of plotter units. If the parameters in the groups 'GRAPH.' and 'GRID.' have their default values and the plotter being used has 1024x1024 addressable positions, then each of these three parameters will be given the value 818.4 = (.95-.15) * 1023.
4.117 "SECONDARY/AXIS."

Simplest form of name: 'SEC/AX.'

A group of eighty parameters having to do with the drawing of the four axes. Subgroups and the number of parameters in each are as follows:

- 'SECONDARY/[AXIS/]LEFT.' (20)
- 'SECONDARY/[AXIS/]RIGHT.' (20)
- 'SECONDARY/[AXIS/]BOTTOM.' (20)
- 'SECONDARY/[AXIS/]TOP.' (20)

The parameters from the subgroups are interleaved in the group; that is to say, the first elements of the subgroups comprise elements 1 through 4 of the group, the second elements of the subgroups comprise elements 5 through 8 of the group, and so on.

4.118 "SECONDARY/[AXIS/]s."

Simplest form of name: 'SEC/s.'

A group of twenty parameters having to do with the drawing of the axis specified by "s", where "s" is one of the keywords "LEFT", "RIGHT", "BOTTOM", or "TOP". Subgroups and the number of parameters in each are as follows:

- 'SECONDARY/[AXIS/]s/POSITION.' (6)
- 'SECONDARY/[AXIS/]s/TICKS.' (3)
- 'SECONDARY/[AXIS/]s/NUMERIC.' (11)

4.119 "SECONDARY/[AXIS/]s/POSITION."

Simplest form of name: 'SEC/s/PO.'

A group of six floating-point numbers, the first three of which describe a point at the beginning of axis "s" and the last three of which describe a point at the end of axis "s". The first two numbers of each triplet are the x and y coordinates of the point, in the grid coordinate system. The third number of each triplet is a user-system x or y coordinate (an x coordinate for a horizontal axis, a y coordinate for a vertical axis) of the point.

4.120 "SECONDARY/[AXIS/]s/TICKS."

Simplest form of name: 'SEC/s/TI.'
A group of three floating-point numbers, specifying the values AUTOGRAPH has chosen to use for the primary parameters

`[AXIS/]s/[TICKS/]MAJOR/[SPACING/]TYPE.`
`[AXIS/]s/[TICKS/]MAJOR/[SPACING/]BASE.`
`[AXIS/]s/[TICKS/]MINOR/SPACING.`

(which see, above). These secondary parameters are used to hold the values AUTOGRAPH chooses for the corresponding primary parameters so as not to disturb "null 1" values of those primary parameters.

4.121 `SECONDARY/[AXIS/]s/NUMERIC.`

Simplest form of name: `SEC/s/NU.`

A group of eleven floating-point numbers having to do with the generation of numeric labels on the axis specified by "s". The first three of these specify the values AUTOGRAPH has chosen to use for the primary parameters

`[AXIS/]s/[NUMERIC/]TYPE.`
`[AXIS/]s/[NUMERIC/]EXponent.`
`[AXIS/]s/[NUMERIC/]FRACTION.`

(which see, above). The secondary parameters are used so as not to disturb "null 1" values of the primary parameters.

The fourth parameter is an integral floating-point count of the number of characters in the longest numeric-label mantissa on the axis "s".

The fifth parameter is an integral floating-point count of the number of characters in the longest numeric-label exponent on the axis "s".

The sixth parameter is the necessary multiplicative "reduction factor" (between "0." and "1." ) to be applied to the sizes of numeric labels on the axis "s" in order to make them fit without overlap problems.

The seventh, eighth, ninth, and tenth parameters are floating-point numbers specifying the width of the space required by numeric labels to the left (outward), to the right (inward), at the beginning and at the end of the axis "s" - each is stated as a fraction of the width or height of the grid window, depending on the orientation of the axis "s".

The eleventh parameter indicates the linear/log nature of the axis specified by "s".

4.122 `SECONDARY/LABEL.`

Simplest form of name: `SEC/LA.`
A group of fifty-four parameters describing the six "label boxes", each of which provides a mechanism for moving and/or shrinking a particular group of labels in attempting to keep any label in that group from overlapping an axis or extending outside the current graph window. Subgroups and the number of parameters in each are as follows:

- SECONDARY/LABEL/LEFT. (9)
- SECONDARY/LABEL/RIGHT. (9)
- SECONDARY/LABEL/BOTTOM. (9)
- SECONDARY/LABEL/TOP. (9)
- SECONDARY/LABEL/CENTER. (9)
- SECONDARY/LABEL/GRAPH. (9)

The parameters of the subgroups are interleaved in the group. The first elements of the subgroups form elements 1 through 6 of the group, the second elements of the subgroups form elements 7 through 12 of the group, and so on.

4.123  "SECONDARY/LABEL/b.".

Simplest form of name: 'SEC/LA/b.'

where the keyword "b" specifies the label box, as follows:

- If "b" = "LEFT", label box 1 is specified. It contains all labels having a basepoint on the left edge of the grid window and a leftward-pointing offset vector. These labels are to be moved leftward as required to avoid overlapping any numeric labels on either y axis.

- If "b" = "RIGHT", label box 2 is specified. It contains all labels having a basepoint on the right edge of the grid window and a rightward-pointing offset vector. These labels are to be moved rightward as required to avoid overlapping any numeric labels on either y axis.

- If "b" = "BOTTOM", label box 3 is specified. It contains all labels having a basepoint on the bottom edge of the grid window and a downward-pointing offset vector. These labels are to be moved downward as required to avoid overlapping any numeric labels on either x axis.

- If "b" = "TOP", label box 4 is specified. It contains all labels having a basepoint on the top edge of the grid window and an upward-pointing offset vector. These labels are to be moved upward as required to avoid overlapping any numeric labels on either x axis.

- If "b" = "CENTER", label box 5 is specified. It contains all labels having a basepoint on some edge of the grid window and an inward-pointing offset vector. These labels are to be moved inward as required to avoid overlapping numeric labels on any axis.

- If "b" = "GRAPH", label box 6 is specified. It contains all labels not specifically assigned to one of the other boxes. These labels are not moved, but may still be shrunk as required to avoid their running outside
the grid window.

Prior to a call to AGSTUP, the nine parameters in this group are undefined. Following an AGSTUP call, but preceding an AGBACK call, they have what I shall call "interim" values. Following an AGBACK call, they have what I shall call "final" values.

The first parameter in the group is a "reduction factor" for the widths of characters in the labels in box "b". This parameter may have the interim value "0.", specifying that no actual value has yet been computed, or "1.", specifying that the user has prohibited shrinkage of labels in box "b" (by giving 'LABEL/CONTROL.' the value "1."). The final value of the reduction factor may be "-1.", specifying that minimum-sized labels were used, but even they led to overlap problems, or a value between "0." and "1.", specifying the actual reduction factor applied when the labels were drawn.

The next four parameters in the group specify the grid-system x coordinates of the left and right edges, and the grid-system y coordinates of the bottom and top edges, of label box "b". The interim values specify the box in which the labels must be made to fit in order to avoid overlap, the final values the box in which the labels were actually made to fit.

The last four parameters in the group specify the grid-system x coordinates of the left and right edges, and the grid-system y coordinates of the bottom and top edges, of the label box "b" which would result if all the labels were reduced to minimum size. The interim values specify an unmoved box, the final values a (possibly) moved box.
5. EXAMPLES

This write-up contains the FORTRAN text and Dicomed output from thirteen jobs using AUTOGRAPH to produce graphs. These graphs illustrate many of the capabilities of AUTOGRAPH.

The duplex character set of PWRITX was used for all labelling. See the paragraph "USE OF PWRITX BY AUTOGRAPH", in the section "OVERVIEW".

Additions to this set of examples will be gratefully accepted. Of particular interest are examples illustrating pitfalls of AUTOGRAPH and the manner in which they may be avoided, examples showing creative solutions to graphing problems which are apt to be encountered frequently by others (examples 9 and 11 are additions of this sort), and, last but not least, examples of graphs which just "look neat".

AUTOGRAPH - 95 - EXAMPLES
5.1 EXAMPLE 1. This example illustrates a simple use of EZY. Note that AUTOGRAPH chose to use exponential left-axis numeric labels, since non-exponential labels would have required more characters.

PROGRAM EXMPL1
C
C Define the data array.
C
REAL YDRA(1001)
C
C Fill the data array.
C
...
DO 101 I=1,1001
   X=FLOAT(I)/20.
   YDRA(I)=10.* (X-1.)*(X-11.)*(X-21.)*(X-31.)*(X-41.)*
   + (X-51.)+2.E7*(FRAN()-.5)
101 CONTINUE
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Draw the graph, using EZY.
C
CALL EZY (YDRA,1001, 'EXAMPLE 1 (EZY)'
C
STOP
C
END
FUNCTION FRAN()
C
C Random-number generator.
C
DATA X / 2.7182818 /
SAVE X
X=AMD (9821.*X+.211327,1.)
FRAN=X
RETURN
END
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
5.2 EXAMPLE 2. This example illustrates a simple use of EZXY. Note that x coordinates used need not be monotonically increasing.

```fortran
PROGRAM EXMPL2
C
C Define the data arrays.
C
REAL XDRA(4001), YDRA(4001)
C
C Fill the data arrays.
C
DO 101 I=1,4001
```

EXAMPLES AUTOGRAPH

- 98 -
THEIA = .0015707963267949*FLOAT(I-1)
RHO = SIN(2.*THETA) + .05*SIN(64.*THETA)
XDRA(I) = RHO*COS(THETA)
YDRA(I) = RHO*SIN(THETA)

101 CONTINUE

C Draw a boundary around the edge of the plotter frame.
C
C CALL BNDARY
C
C Draw the graph, using EZXY.
C
C CALL EZXY (XDRA,YDRA,4001, ´EXAMPLE 2 (EZXY)´)
C
C STOP
C
C END
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
5.3 **EXAMPLE 3.** This example illustrates a simple use of EZMY. A remarkably uninteresting graph.

**EXAMPLE 3 (EZMY)**

```
PROGRAM EXMPL3
  C Define the data array.
  C
  REAL YDRA(100,2)
  C Fill the data array.
  C
  DO 101 I=1,100
```

```
YDRA(I,1) = \cos(3.14159265358979 \times \text{FLOAT}(I)/25.) \times \\
+ \text{FLOAT}(I)^2
YDRA(I,2) = \cos(3.14159265358979 \times \text{FLOAT}(I)/25.) \times \\
+ 10. \times (0.04 \times \text{FLOAT}(I))

101 \text{ CONTINUE}

C
C Draw a boundary around the edge of the plotter frame.
C
C \text{CALL BNDARY}
C
C Draw the graph, using \text{EZMY}.
C
C \text{CALL EZMY (YDRA,100,2,100, \text{"EXAMPLE 3 (EZMY)$"})}
C
C \text{STOP}
C
C \text{END}

\text{SUBROUTINE BNDARY}

C
C Routine to draw the plotter-frame edge.
C
C \text{CALL PLOTIT ( 0, 0,0)}
C \text{CALL PLOTIT (32767, 0,1)}
C \text{CALL PLOTIT (32767,32767,1)}
C \text{CALL PLOTIT ( 0,32767,1)}
C \text{CALL PLOTIT ( 0, 0,1)}
C \text{RETURN}
C \text{END}
5.4 EXAMPLE 4. This example illustrates a simple use of EZMXY - nested ellipses.

```
PROGRAM EXMPL4
C
C Define the data arrays.
C
REAL XDRA(201),YDRA(201,10)
C
C Fill the data arrays.
C
DO 102 I=1,201
```

EXAMPLE 4 (EZMXY)
XDRA(I) =-1.+0.02*FLOAT(I-1)
IF (I.GT.101) XDRA(I)=2.-XDRA(I)
DO 101 J=1,10
   YDRA(I,J)=FLOAT(J)*
   + SQRT(1.000000000001-XDRA(I)**2)/10.
   IF (I.GT.101) YDRA(I,J)=-YDRA(I,J)
101 CONTINUE
102 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)$')
C
STOP
C
END
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
5.5 **EXAMPLE 5.** This example illustrates a relatively simple use of EZMXY; generating the data requires using some really fascinating numeric constants. Essentially, six circles are drawn; two small portions of each are blanked out by salting in some "null 1's" (1.E36's). The result is a possibly recognizable commercial logo.

```
EXAMPLE 5 (EZMXY)
```

```
PURITY, BODY, AND FLAVOR
```

```
PROGRAM EXMPL5
C
C Define the data arrays.
C
REAL XDRA(401,6), YDRA(401,6)
C
```
C Compute required constants.
C
PI=3.14159265358979
PID200=PI/200.
PITW=2.*PI
PIT2D3=2.*PI/3.
PIT4D3=4.*PI/3.
RADOSC=SQRT(3.)/3.
RADOLC=SQRT(3.)/2.
BSSC=ATAN(SQRT(12.)/6.)
BSSCUL=ATAN(SQRT(143.)/7.)
BSLCLL=ATAN(SQRT(143.)/17.)
BSLCUL=ATAN(SQRT(2.0))

C Fill the data arrays.
C
DO 101 I=1,401
THETA=PID200*FLOAT(I-1)
XDRA(I,1)= -.5+RADOSC*COS(THETA)
YDRA(I,1)= RADOSC*SIN(THETA)
IF (ABS(THETA ) .GE.BSSCLL.AND.
+ ABS(THETA-PITIWO) .GE.BSSCUL) XDRA(I,1)=1.E36
IF (ABS(THETA-PIT2D3) .GE.BSSCUL) XDRA(I,1)=1.E36
XDRA(I,2)= .5+RADOSC*COS(THETA)
YDRA(I,2)= RADOSC*SIN(THETA)
IF (ABS(THETA-PIT2D3) .GE.BSSCLL) XDRA(I,1)=1.E36
XDRA(I,2)= RADOLC*RADOIC*SIN(THETA)
IF (ABS(THETA-PIT4D3) .GE.BSSCLL.AND.
+ ABS(THETA-PIT4D3) .GE.BSSCUL) XDRA(I,1)=1.E36
XDRA(I,3)= RADOLC*COS(THETA)
YDRA(I,3)= RADOLC*SIN(THETA)
IF (ABS(THETA-PIT4D3) .GE.BSSCUL) XDRA(I,1)=1.E36

101 CONTINUE
C
C Specify subscripting of XDRA and YDRA.
C
CALL AGSETI (ROW.',2)
C
C Set up grid shape to make 1 unit in x = 1 unit in y.
CALL AGSETF ('GRID/SHAPE.', 2.)

Thrn off background, then turn labels back on.

CALL AGSETF ('BACKGROUND.', 4.)
CALL AGSETI ('LABEL/CONTROL.', 2)

Turn off left label.

CALL AGSETC ('LABEL/NAME.', 'L')
CALL AGSETI ('LABEL/SUPPRESSION FLAG.', 1)

Change text of bottom label.

CALL AGSETC ('LABEL/NAME.', 'B')
CALL AGSETI ('LINE/NUMBER.', -100)
CALL AGSETC ('LINE/TEXT.', 'PURITY, BODY, AND FLAVORS')

Draw a boundary around the edge of the plotter frame.

CALL BNDARY

Draw the graph, using EZMXY.

CALL EZMXY (XDRA, YDRA, 401, 6, 401, 'EXAMPLE 5 (EZMXY) $

STOP

END

SUBROUTINE BNDARY

Routine to draw the plotter-frame edge.

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
5.6 EXAMPLE 6. This example illustrates the use of the graph window and the four principal types of backgrounds one can use: perimeter, grid, half-axis, and "none". Note that I have cheated a bit and turned the labels back on for the last of these. This example also illustrates the use of linear and logarithmic x and y mappings.

PROGRAM EXAMPLE6
C Define the data arrays.
C
REAL XDRA(501), YDRA(501)
C
AUTOGRAF
C Define the graph-window parameter array.
C
REAL GWND (4,4)
C
DATA (GWND(I,1),I=1,4) / 0.0 , 0.5 , 0.5 , 1.0 /
DATA (GWND(I,2),I=1,4) / 0.5 , 1.0 , 0.5 , 1.0 /
DATA (GWND(I,3),I=1,4) / 0.0 , 0.5 , 0.0 , 0.5 /
DATA (GWND(I,4),I=1,4) / 0.5 , 1.0 , 0.0 , 0.5 /
C
C Define variables used in setting up informational labels.
C
CHARACTER*35 GLAB
CHARACTER*23 BACK(4)
CHARACTER*12 LNLG(4)
C
DATA BACK(1) / '(PERIMETER BACKGROUND)$' / 
DATA BACK(2) / '(GRID BACKGROUND)$' / 
DATA BACK(3) / '(HALF-AXIS BACKGROUND)$' / 
DATA BACK(4) / '(NO BACKGROUND)$' / 
C
DATA LNLG(1) / 'LINEAR$' / 
DATA LNLG(2) / 'LOGARITHMICS$' / 
C
C Fill the data arrays.
C
DO 101 I=1,501
  IHETA=.031415926535898*FLOAT(I-1)
  XDRA(I) =500.+.9*FLOAT(I-1)*COS(IHETA)
  YDRA(I) =500.+.9*FLOAT(I-1)*SIN(IHETA)
101 CONTINUE
C
C Suppress the frame advance.
C
CALL AGSETI ('FRAME.' ,2)
C
C Do four graphs on the same frame, using different
C backgrounds.
C
DO 102 IGRF = 1,4
C
C Position the graph window.
C
CALL AGSETP ('GRAPH WINDOW.' ,GWND(1,IGRF) ,4)
C
C Declare the background type.
C
CALL AGSETI ('BACKGROUND TYPE.' ,IGRF)
C
C Setting the background type may have turned informational
C labels off. In that case, turn them back on.
C
IF (IGRF.EQ.4) CALL AGSETI ('LABEL/CONTROL.' ,2)
C Set up parameters determining linear/log nature of axes.
C
    ILLX=(IGRF-1)/2
    ILLY=MOD(IGRF-1,2)
C
C Declare the linear/log nature of the graph.
C
    CALL AGSETI ('X/LOGARITHMIC.', ILLX)
    CALL AGSETI ('Y/LOGARITHMIC.', ILLY)
C
C Change the x- and y-axis labels to reflect the linear/log
C nature of the graph.
C
    CALL AGSETC ('LABEL/NAME.', 'B')
    CALL AGSETI ('LINE/NUMBER.', -100)
    CALL AGSETC ('LINE/TEXT.', INLG(ILLX+1))
C
    CALL AGSETC ('LABEL/NAME.', 'L')
    CALL AGSETI ('LINE/NUMBER.', 100)
    CALL AGSETC ('LINE/TEXT.', INLG(ILLY+1))
C
C Set up the label for the top of the graph.
C
    WRITE (GLAB,1001) IGRF, BACK(IGRF)
C
C Draw the graph, using EZXY.
C
    CALL EZXY (XDRA,YDRA,501,GLAB)
C
    102 CONTINUE
C
C Draw a boundary around the edge of the plotter frame.
C
    CALL BNDRY
C
C Advance the frame.
C
    CALL FRAME
C
    STOP
C
C Format for encode.
C
    1001 FORMAT ('EXAMPLE 6-', I1, ' ', A23)
END

SUBROUTINE BNDRY
C
C Routine to draw the plotter-frame edge.
C
    CALL PLOTTIT ( 0, 0,0)
    CALL PLOTTIT (32767, 0,1)
    CALL PLOTTIT (32767,32767,1)
    CALL PLOTTIT ( 0,32767,1)

AUTOGRAPH    - 109 -
EXAMPLES
CALL PLOTIT (0, 0, 1)
RETURN
END
5.7 **EXAMPLE 7.** This example illustrates several features. It shows how to define informational labels. It shows how label coordinate systems along an axis are defined. Windowing is used to prevent the curves from running wild. The "user" set of dashed-line patterns is employed. ("Incrudescence" is a word that I invented, by the way.)

![Graph of Lines of Constant Incrudescence]

**EXAMPLE 7 (EZMXY)**

**LINES OF CONSTANT INCRUDESCENCE**

**DISTANCE FROM EQUATOR (MILES)**

![Graph of Lines of Constant Incrudescence]

**HEIGHT (KILOMETERS)**

**PRESSURE (TONS/SQUARE FURLONG)**

**PROGRAM EXMPL7**

C
C Define the data arrays and the dash-pattern array.
C
REAL XDRA(101),YDRA(101,9)
CHARACTER*28 DSHP(9)
C Declare the type of the dash-pattern-name generator.
CHARACTER*16 AGSHN
C Fill the data arrays and the dash pattern array.
DO 101 I=1,101
   XDRA(I)=-90.+1.8*FLOAT(I-1)
101 CONTINUE
DO 103 J=1,9
   WRITE (DSHP(J),1001) J
   FJ=J
   DO 102 I=1,101
      YDRA(I,J)=3.*FJ-(FJ/2700.)*XDRA(I)**2
102 CONTINUE
103 CONTINUE
C Turn on windowing.
CALL AGSET ('WINDOWING. ',1)
C Move the edges of the curve window (grid).
CALL AGSETF ('GRID/LEFT. ',.10)
CALL AGSETF ('GRID/RIGHT. ',.90)
CALL AGSETF ('GRID/BOTTOM. ',.10)
CALL AGSETF ('GRID/TOP. ',.85)
C Set the x and y minimum and maximum.
CALL AGSETF ('X/MINIMUM. ',-90.)
CALL AGSETF ('X/MAXIMUM. ',+90.)
CALL AGSETF ('Y/MINIMUM. ',0.)
CALL AGSETF ('Y/MAXIMUM. ',18.)
C Set left axis parameters.
CALL AGSETI ('LEFT/MAJOR/TYPE. ',1)
CALL AGSETF ('LEFT/MAJOR/BASE. ',3.)
CALL AGSETI ('LEFT/MINOR/SPACING. ',2)
C Set right axis parameters.
CALL AGSETI ('RIGHT/FUNCTION. ',1)
CALL AGSETF ('RIGHT/Numeric/TYPE. ',1.E36)
C Set bottom axis parameters.
CALL AGSETI ('BOTTOM/MAJOR/TYPE. ',1)
CALL AGSETF ('BOTTOM/MAJOR/BASE. ',15.)
CALL AGSETI ('BOTTOM/MINOR/SPACING. ',2)
C Set top axis parameters.
C
CALL AGSETI ('TOP/FUNCTION.', 2)
CALL AGSETF ('TOP/NUMERIC/TYPEx.,1.E36)
C
C Set up the dash patterns to be used.
C
CALL AGSETI ('DASH/SELECTOR.', 9)
CALL AGSETI ('DASH/LENGTH.', 28)
DO 104 I=1,9
   CALL AGSETC (AGDShN(I), DSHP(I))
104 CONTINUE
C
C Set up the left label.
C
CALL AGSETC ('LABEL/NAME.', 'L')
CALL AGSETI ('LINE/NUMBER.', 100)
CALL AGSETC ('LINE/TEXT.', 'HEIGHT (KILOMETERS)')
C
C Set up the right label.
C
CALL AGSETC ('LABEL/NAME.', 'R')
CALL AGSETI ('LINE/NUMBER.', -100)
CALL AGSETC ('LINE/TEXT.', +
   'PRESSURE (TONS/SQUARE FURLONG)')
C
C Set up the bottom labels.
C
CALL AGSETC ('LABEL/NAME.', 'B')
CALL AGSETI ('LINE/NUMBER.', -100)
CALL AGSETC ('LINE/TEXT.', 'LATITUDE (DEGREES)')
C
CALL AGSETC ('LABEL/NAME.', 'SP')
CALL AGSETF ('LABEL/BASEPOINT/X.', .000001)
CALL AGSETF ('LABEL/BASEPOINT/Y.', 0.)
CALL AGSETF ('LABEL/OFFSET/Y.', -.015)
CALL AGSETI ('LINE/NUMBER.', -100)
CALL AGSETC ('LINE/TEXT.', 'SP$')
C
CALL AGSETC ('LABEL/NAME.', 'NP')
CALL AGSETF ('LABEL/BASEPOINT/X.', .999999)
CALL AGSETF ('LABEL/BASEPOINT/Y.', 0.)
CALL AGSETF ('LABEL/OFFSET/Y.', -.015)
CALL AGSETI ('LINE/NUMBER.', -100)
CALL AGSETC ('LINE/TEXT.', 'NP$')
C
C Set up the top label.
C
CALL AGSETC ('LABEL/NAME.', 'T')
CALL AGSETI ('LINE/NUMBER.', 80)
CALL AGSETC ('LINE/TEXT.', +
   'DISTANCE FROM EQUATOR (MILES)')
CALL AGSETI ('LINE/NUMBER.','90')
CALL AGSETC ('LINE/TEXT.','$')
CALL AGSETI ('LINE/NUMBER.','100')
CALL AGSETC ('LINE/TEXT.','
+ 'LINES OF CONSTANT INCRUDESCENCE$'
) CALL AGSETI ('LINE/NUMBER.','110')
CALL AGSETC ('LINE/TEXT.','EXAMPLE 7 (EZMXY)$')

C Set up centered (box 6) label.
C CALL AGSETI ('LABEL/NAME.','EQUATOR')
CALL AGSETI ('LABEL/ANGLE.','90')
CALL AGSETI ('LINE/NUMBER.','0')
CALL AGSETC ('LINE/TEXT.','EQUATOR$')

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,101,9,101,0)
C
STOP
C
C Format for encode above.
C 1001 FORMAT ('$$$$$$$$$$$$$$$$$$$$$J''=',''Il,''')
C
END
SUBROUTINE AGUTOL (IAXS,FUNS,IDMA,VINP ,VOTP)
C
C Mapping for the right axis.
C IF (FUNS.EQ.1.) THEN
IF (IDMA.GT.0) VOTP=ALOG10(20.-VINP)
IF (IDMA.LT.0) VOTP=20.-10.**VINP
C
C Mapping for the top axis.
C ELSE IF (FUNS.EQ.2.) THEN
IF (IDMA.GT.0) VOTP=70.136*VINP
IF (IDMA.LT.0) VOTP=VINP/70.136
C
C Default (identity) mapping.
C ELSE
VOTP=VINP
END IF
C
Done.
C RETURN

EXAMPLES - 114 -
C
END

SUBROUTINE BNDARY

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
5.8 EXAMPLE 8. This example is a somewhat ugly graph demonstrating that one can plot "x as a function of y", that label values can run "backwards" along an axis, and that axes need not have major tick marks at their ends. The "alphabetic" set of dashed-line patterns is used. Major tick marks on the x axis are extended in both directions.

PROGRAM EXMPL8
C
C Define the data arrays.
C
REAL XDRA(10), YDRA(4,101)
C Fill the data arrays.
C
DO 101 I=1,101
   XDRA(I) = -3.14159265358979 +
      .062831853071796*FLOAT(I-1)
101 CONTINUE

C
DO 103 I=1,4
   FLTI=I
   BASE=2.*FLTI-1.
   DO 102 J=1,101
      YDRA(I,J) = BASE + .75*SIN(-3.14159265358979 +
         .062831853071796*FLTI*FLOAT(J-1))
102 CONTINUE
103 CONTINUE

C Change the line-end character to a period.
C
CALL AGSETC ('LINE/END.„,„')
C
C Specify labels for x and y axes.
C
CALL ANOTAT ('SINE FUNCTIONS OF T.„,„,0,0,0,0')
C
C Use a half-axis background.
C
CALL AGSETI ('BACKGROUND.„,3')
C
C Move x axis to the zero point on the y axis.
C
CALL AGSETF ('BOTTOM/INTERSECTION/USER.„,0.)
C
C Specify base value for spacing of major ticks on x axis.
C
CALL AGSETF ('BOTTOM/MAJOR/BASE.„,1.)
C
C Run major ticks on x axis to edge of curve window.
C
CALL AGSETF ('BOTTOM/MAJOR/INWARD.„,1.)
CALL AGSETF ('BOTTOM/MAJOR/OUTWARD.„,1.)
C
C Position x axis minor ticks.
C
CALL AGSETI ('BOTTOM/MINOR/SPACING.„,9)
C
C Run the y axis backward.
C
CALL AGSETI ('Y/ORDER.„,1)
C
C Run plots full-scale in y.
C
CALL AGSETI ('Y/NICE.„,0')
C Have AUTOGRAPH scale x and y data the same.
C
CALL AGSETF ('GRID/SHAPE.',.01)
C
Use the alphabetic set of dashed-line patterns.
C
CALL AGSETI ('DASH/SELECTOR.',-1)
C
Tell AUTOGRAPH how the data arrays are dimensioned.
C
CALL AGSETI ('ROW.',-1)
C
Reverse the roles of the x and y arrays.
C
CALL AGSETI ('INVERT.',1)
C
Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
Draw the curves.
C
CALL EZMXY (XDRA,YDRA,4,4,101,'EXAMPLE 8.')
C
STOP
C
END
SUBROUTINE BNDARY
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
5.9 EXAMPLE 9. This example shows how to create a hybrid logarithmic axis with positive values on one end and negative values on the other; a line of discontinuity separates the two. The routine AGCHNL is replaced by a version which forces the numeric label in the middle of the left axis to be blank.

```
PROGRAM EXMPL9
C
C Define the data arrays.
C
DIMENSION XDAT(400), YDAT(400)
C
C Fill the data arrays.
```

AUTOGRAPH

- 119 -

EXAMPLES
DO 101 I=1,400
   XDAT(I)=(FLOAT(I)-1.)/399.
101 CONTINUE

CALL GENDAT (YDAT( 1),200,200,1,3,3,+.01,+10.)
CALL GENDAT (YDAT(201),200,200,1,3,3,-10.,-.01)

C The y data ranges over both positive and negative values.
C It is desired that both ranges be represented on the same
C graph and that each be shown logarithmically, ignoring
C values in the range -.01 to +.01, in which we have no
C interest. First we map each y datum into its absolute
C value (.01 if the absolute value is too small). Then we
C take the base-10 logarithm, add 2.0001 (so as to be sure
C of getting a positive number), and re-attach the original
C sign. We can plot the resulting y data on a linear y axis.

DO 102 I=1,400
   YDAT(I)=SIGN(ALOG10(AMAX1(ABS (YDAT(I)),.01))+2.0001,
               + YDAT(I))
102 CONTINUE

C In order that the labels on the y axis should show the
C original values of the y data, we change the user-system-
C to-label-system mapping on both y axes and force major
C ticks to be spaced logarithmically in the
C label system (which will be defined by the subroutine
C AGUTOL in such a way as to re-create numbers in the
C original range).

CALL AGSETI (´LEFT/FUNCTION.´,1)
CALL AGSETI (´LEFT/MAJOR/TYP.´,2)

CALL AGSETI (´RIGHT/FUNCTION.´,1)
CALL AGSETI (´RIGHT/MAJOR/TYP.´,2)

C Change the left-axis label to reflect what's going on.

CALL AGSETC (´LABEL/NAME.´,´L´)
CALL AGSETI (´LINE/NUMBER.´,100)
CALL AGSETC (´LINE/TEXT.´,
            +  ´LOG SCALING, POSITIVE AND NEGATIVE$´)

C Draw a boundary around the edge of the plotter frame.

CALL BNDARY

C Draw the curve.

CALL EZXY (XDAT,YDAT,400,´EXAMPLE 9$´)

STOP
SUBROUTINE ?ALUT (IAXS,FUNS, IDMA,VINP,VOTP)

C Left or right axis.
C
IF (FUNS.EQ.1.) THEN
  IF (IDMA.LT.0) THEN
    VOTP=SIGN(10.**(ABS(VINP)-2.0001),VINP)
  ELSE
    VOTP=SIGN(10.**(ABS(VINP)-2.0001),VINP)
  END IF
C
C All others.
C
ELSE
  VOTP=VINP
END IF
C
C Done.
C
RETURN
C
END SUBROUTINE AGCHNL (IAXS, VILS, CHRM, MCIM, NCIM, IPXM, +
  CHRE, MCIE, NCIE)

C
C CHARACTER*(*) CHRM,CHRE
C
C Modify the left-axis numeric label marking the value "0."
C
IF (IAXS.EQ.1 .AND. VILS.BG.0.) THEN
  CHRM(1:1)=' '
  NCIM=1
  IPXM=0
  NCIE=0
END IF
C
C Done.
C
RETURN
C
END SUBROUTINE GENDAT (DATA, IDIM, M, N, MLOW, MHGH, DLOW, DHGH)

C
C This is a routine to generate test data for two-dimensional
C graphics routines. Given an array "DATA", dimensioned
C "IDIM,1", it fills the sub-array ((DATA(I,J),I=1,M),J=1,N)
C with a two-dimensional field of data having approximately
C "MLOW" lows and "MHGH" highs, a minimum value of exactly
C "DLOW" and a maximum value of exactly "DHGH".
C
C "MLOW" and "MHGH" are each forced to be greater than or
C equal to $1$ and less than or equal to $25$.
C The function used is a sum of exponentials.
C
\begin{verbatim}
DIMENSION DATA(IDIM,1),CCNT(3,50)

FOVM=9./FLOAT(M)
FOVN=9./FLOAT(N)

NLOW=MAX0(1,MIN0(25,MLOW))
NHGH=MAX0(1,MIN0(25,MHIGH))
NCNT=NLOW+NHIGH

DO 101 K=1,NCNT
   CCNT(1,K)=1.+(FLOAT(M)-1.)*FRAN()
   CCNT(2,K)=1.+(FLOAT(N)-1.)*FRAN()
   IF (K.LE.NLOW) THEN
      CCNT(3,K)=-1.
   ELSE
      CCNT(3,K)=+1.
   END IF
101 CONTINUE

DMIN=+1.E36
DMAX=-1.E36
DO 104 J=1,N
   DO 103 I=1,M
      DATA(I,J)=.5*(DLOW+DHGH)
      DO 102 K=1,NCNT
         DATA(I,J)=DATA(I,J)+.5*(DHGH-DLOW)*CCNT(3,K)*
         + EXP(-((FOVM*(FLOAT(I)-CCNT(1,K)))**2+
         + (FOVN*(FLOAT(J)-CCNT(2,K)))**2))
      102 CONTINUE
      DMIN=AMIN1(DMIN,DATA(I,J))
      DMAX=AMAX1(DMAX,DATA(I,J))
   103 CONTINUE
104 CONTINUE

DO 106 J=1,N
   DO 105 I=1,M
      DATA(I,J)=(DATA(I,J)-DMIN)/(DMAX-DMIN)*(DHGH-DLOW)
      + (DLOW)
   105 CONTINUE
106 CONTINUE

RETURN
END

FUNCTION FRAN()

C Random-number generator.
C
DATA X / 2.7182818 /
\end{verbatim}

\textbf{EXAMPLES} - 122 -
\textbf{AUTOGRAPH}
SAVE X
X=AMOD(9821.*X+.211327,1.)
FRAN=X
RETURN
END

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
5.10 EXAMPLE 10. This example shows how to create non-standard numeric labels on the axes. The bottom x axis is labelled with the names of the months, and the left y axis with Roman numerals.

```
PROGRAM XMPL10
C
C Define the data arrays.
C
REAL XDRA(1201), YDRA(1201)
C
C Fill the data arrays. The independent variable represents
C time during the year (a hypothetical year with equal-length
```
C months) and is set up so that minor ticks can be lengthened
to delimit the months; the major ticks, though shortened
to invisibility, still determine where the labels go.

C

DO 101 I=1,1201
   XDRA(I)=FLOAT(I-51)
   YDRA(I)=COSH(FLOAT(I-601)/202.)
101 CONTINUE
C
C Change the labels on the bottom and left axes.
C
CALL ANOTAT ('MONTHS OF THE YEAR$','
               + 'ROMAN NUMERALS$\',0,0,0,0)
C
C Fix the minimum and maximum values on both axes and prevent
AUTOGRAPH from using rounded values at the ends of the axes.

C
CALL AGSETF ('X/MIN.',-50.)
CALL AGSETF ('X/MAX.',1150.)
CALL AGSETI ('X/NICE.',0)

C
CALL AGSETF ('Y/MIN.',1.)
CALL AGSETF ('Y/MAX.',10.)
CALL AGSETI ('Y/NICE.',0)

C Specify the spacing between major tick marks on all axes.
C Note that the AUTOGRAPH dummy routine AGCHNL is supplanted
C (below) by one which supplies dates for the bottom axis and
C Roman numerals for the left axis in place of the numeric
C labels one would otherwise get.

C
CALL AGSETI ('LEFT/MAJOR/TYE.',1)
CALL AGSETI ('RIGHT/MAJOR/TYE.',1)
CALL AGSETI ('BOTTOM/MAJOR/TYE.',1)
CALL AGSETI ('TOP/MAJOR/TYE.',1)

C
CALL AGSETF ('LEFT/MAJOR/BASE.',1.)
CALL AGSETF ('RIGHT/MAJOR/BASE.',1.)
CALL AGSETF ('BOTTOM/MAJOR/BASE.',100.)
CALL AGSETF ('TOP/MAJOR/BASE.',100.)

C Suppress minor ticks on the left and right axes.

C
CALL AGSETI ('LEFT/MINOR/SPACING.',0)
CALL AGSETI ('RIGHT/MINOR/SPACING.',0)

C On the bottom and top axes, put one minor tick between each
C pair of major ticks, shorten major ticks to invisibility,
C and lengthen minor ticks. The net effect is to make the
C minor ticks delimit the beginning and end of each month,
C while the major ticks, though invisible, cause the names of
C the months to be where we want them.

C

AUTOGRAPH - 125 - EXAMPLES
CALL AGSETI ("BOTTOM/MINOR/SPACING.",1)
CALL AGSETI ("TOP/MINOR/SPACING.",1)

CALL AGSETF ("BOTTOM/MAJOR/INWARD.",0.)
CALL AGSETF ("BOTTOM/MINOR/INWARD.",0.015)
CALL AGSETF ("TOP/MAJOR/INWARD.",0.)
CALL AGSETF ("TOP/MINOR/INWARD.",0.015)

C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Draw the graph, using EZXY.
C
CALL EZXY (XDRA,YDRA,1201,
        'EXAMPLE 10 (MODIFIED NUMERIC LABELS)')

STOP
C
END
SUBROUTINE AGCHNL (IAXS,VILS,CHR,M,NCIM,IPXM,
        CHRE,NCIE,NCIE)
C CHARACTER(*) CHRM,CHRE

C Define the names of the months for use on the bottom axis.
C
CHARACTER*3 MONS(12)
DATA MONS / 'JAN','FEB','MAR','APR','MAY','JUN',
        'JUL','AUG','SEP','OCT','NOV','DEC'/

C Modify the numeric labels on the left axis.
C
IF (IAXS.EQ.1) THEN
    CALL AGORN (IFIX(VILS),CHR,N)
    IPXM=0
    NCIE=0
ENDIF

C Modify the numeric labels on the bottom axis.
C
ELSEIF (IAXS.EQ.3) THEN
    IMON=IFIX(VILS+.5)/100+1
    CHRM(1:3)=MONS(IMON)
    NC:-3
    I:0
    NCIE=0
ENDIF

C Done.
C
RETURN
C
END
SUBROUTINE AGCORN (NTGR, BCRN, NCRN)
C
CHARACTER*(*) BCRN
C
This routine receives an integer in NTGR and returns its
Roman-numeral equivalent in the first NCRN characters of
the character variable BCRN. It only works for integers
within a limited range and it does some rather unorthodox
things (like using zero and minus).
C
ICH1, ICH5, and IC10 are character variables used for the
single-unit, five-unit, and ten-unit symbols at a given
level.
C
CHARACTER*1 ICH1, ICH5, IC10
C
Treat numbers outside the range (-4000,+4000) as infinites.
C
IF (IABS (NTGR).GE.4000) THEN
  IF (NTGR.GT.0) THEN
    NCRN=5
    BCRN(1:5)=’(INF)’
  ELSE
    NCRN=6
    BCRN(1:6)=’(-INF)’
  END IF
  RETURN
END IF
C
Use a ‘0’ for the zero. The Romans never had it so good.
C
IF (NTGR.EQ.0) THEN
  NCRN=1
  BCRN(1:1)=’0’
  RETURN
END IF
C
Zero the character counter.
C
NCRN=0
C
Handle negative integers by prefixing a minus sign.
C
IF (NTGR.LT.0) THEN
  NCRN=NCRN+1
  BCRN(NCRN:NCRN)=’-’
END IF
C
Initialize constants. We’ll check for thousands first.
C
IMDD=10000
IDIV=1000
ICH1=’M’
Find out how many thousands (hundreds, tens, units) there are and jump to the proper code block for each case.

101 INTG=MOD(IABS(NTGR),IMOD)/IDIV

GO TO (107,104,104,104,102,103,103,103,106),INTG+1

Four - add ICH1 followed by ICH5.

102 NCRN=NCRN+1
   BCRN(NCRN:NCRN)=ICH1

Five through eight - add ICH5, followed by INTG-5 ICH1's.

103 NCRN=NCRN+1
   BCRN(NCRN:NCRN)=ICH5

   INTG=INTG-5
   IF (INTG.LE.0) GO TO 107

One through three - add that many ICH1's.

104 DO 105 I=1,INTG
   NCRN=NCRN+1
   BCRN(NCRN:NCRN)=ICH1
105 CONTINUE

GO TO 107

Nine - add ICH1, followed by IC10.

106 NCRN=NCRN+1
   BCRN(NCRN:NCRN)=ICH1
   NCRN=NCRN+1
   BCRN(NCRN:NCRN)=IC10

If we're done, exit.

107 IF (IDIV.EQ.1) RETURN

Otherwise, tool up for the next digit and loop back.

IMOD=IMOD/10
IDIV=IDIV/10
IC10=ICH1

IF (IDIV.EQ.100) THEN
   ICH5="D"
   ICH1="C"
ELSE IF (IDIV.EQ.10) THEN
   ICH5="L"
   ICH1="X"

EXAMPLES - 128 - AUTOGRAPH
ELSE
  ICH5 = 'V'
  ICH1 = 'I'
END IF
C
GO TO 101
C
END
SUBROUTINE BNDARY
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
5.11 EXAMPLE 11. This example shows how to create a scattergram.

PROGRAM XMPL11
C
C Create a scattergram.
C
REAL XDRA(500), YDRA(500)
C
C Fill the data arrays.
C
DO 101 I=1,500
   XDRA(I)=.5+(2.*(FRAN()-0.5))**5
YDRA(I) = .5 + (2.*(FRAN() - .5))**5
101 CONTINUE
C
C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Suppress the frame advance.
C
CALL AGSETI ("FRAME.", 2)
C
C Suppress the drawing of curves by the EZ... routines.
C
CALL AGSETI ("SET.", -1)
C
C Draw the background, using EZXY.
C
CALL EZXY (XDRA, YDRA, 500, "EXAMPLE 11 (SCATTERGRAM) $")
C
C Put a plus sign at each of the x-y positions.
C
CALL POINTS (XDRA, YDRA, 500, "+", 0)
C
C Advance the frame.
C
CALL FRAME
C
STOP
C
END

FUNCTION FRAN()

C Random-number generator.
C
DATA X / 2.7182818 /
SAVE X
X = AMOD(9821.*X+.211327,1.)
FRAN = X
RETURN
END

SUBROUTINE BNDARY

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
5.12 EXAMPLE 12. This example shows how to create the effect of a histogram. Shading is done by an XLIB utility called FILL.

```
PROGRAM XMPL12
C
C Create a sort of histogram.
C
C REAL XDRA(249),YDRA(249),WORK(204),IWRK(204)
C
C Fill the data arrays. First, we define the histogram outline. This will be used in the call to FILL which fills in the area under the histogram.
```
XDRA(1) = 0.
YDRA(1) = 0.

DO 101 I = 2, 100, 2
   XDRA(I) = XDRA(I-1)
   YDRA(I) = EXP(-16.* (FLOAT(I/2)/50.-.51)**2)+.1*FRAN()
   XDRA(I+1) = XDRA(I-1)+.02
   YDRA(I+1) = YDRA(I)
101 CONTINUE

XDRA(102) = 1.
YDRA(102) = 0.

C Define lines separating vertical boxes from each other.
C
NDRA = 102

DO 102 I = 3, 99, 2
   XDRA(NDRA+1) = 1.E36
   YDRA(NDRA+1) = 1.E36
   XDRA(NDRA+2) = XDRA(I)
   YDRA(NDRA+2) = 0.
   XDRA(NDRA+3) = XDRA(I)
   YDRA(NDRA+3) = AMIN1(YDRA(I), YDRA(I+1))
   NDRA = NDRA+3
102 CONTINUE

C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Suppress the frame advance.
C
CALL AGSETI ("FRAME.", 2)
C
C Draw the graph, using EZXY.
C
CALL EZXY (XDRA, YDRA, 249, "EXAMPLE 12 (HISTOGRAM)"
C
C Use the XLIB routine FILL to fill the area defined by the
C data. Note that FILL is not part of the AUTOGRAPH package.
C
CALL FILLOP ("AN", 45)
CALL FILLOP ("SP", 128)
CALL FILL (XDRA, YDRA, 102, WORK, 204, IWRK, 204)
C
C Advance the frame.
C
CALL FRAME
C
STOP
FUNCTION FRAN()
Random-number generator.
DATA X / 2.7182818 /
SAVE X
X=AMOD(9821.*X+.211327,1.)
FRAN=X
RETURN
END

SUBROUTINE BNDARY
Routine to draw the plotter-frame edge.
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
5.13 A FINAL EXAMPLE. The final example conveys an appropriate message. It demonstrates two of the peculiar types of numeric labels which may be generated. (There are others.) It also demonstrates the use of dashed minor ticks (a hellishly time-consuming business which sometimes produces a nice effect).

```
LOGARITHMIC, BASE 2, EXPONENTIAL LABELS

PROGRAM EXMPLF
C
C Define the data array.
C
DIMENSION XYCD(226)
C
```
C Fill the data array.
C
READ 1001, XYCD
C
DO 101 I=1,226
   IF (XYCD(I).EQ.1.E36) GO TO 101
   XYCD(I)=2.**(XYCD(I)-15.)/2.5)
101 CONTINUE
C
C Specify log/log plot.
C
CALL DISPLA (0,0,4)
C
C Bump the line-maximum parameter past 42.
C
CALL AGSETI ('LINE/MAXIMUM.',50)
C
C Specify x- and y-axis labels, grid background.
C
   CALL ANOTAT('LOGARITHMIC, BASE 2, EXPONENTIAL LABELS$+',
              'LOGARITHMIC, BASE 2, NO-EXPONENT LABELS$+',
              2,0,0,0)
C
C Specify the graph label.
C
   CALL AGSETC ('LABEL/NAME.','T')
   CALL AGSETI ('LINE/NUMBER.',100)
   CALL AGSETC ('LINE/TEXT.','FINAL EXAMPLES')
C
C Specify x-axis ticks and labels.
C
   CALL AGSETI ('BOTTOM/MAJOR/TYPES',3)
   CALL AGSETF ('BOTTOM/MAJOR/BASE.',2.)
   CALL AGSETI ('BOTTOM/NUMERIC/TYPES',3)
   CALL AGSETI ('BOTTOM/MINOR/SPACING.',4)
   CALL AGSETI ('BOTTOM/MINOR/PATTERN.',125252B)
C
C Specify y-axis ticks and labels.
C
   CALL AGSETI ('LEFT/MAJOR/TYPES',3)
   CALL AGSETF ('LEFT/MAJOR/BASE.',2.)
   CALL AGSETI ('LEFT/NUMERIC/TYPES',3)
   CALL AGSETI ('LEFT/MINOR/SPACING.',4)
   CALL AGSETI ('LEFT/MINOR/PATTERN.',125252B)
C
C Compute secondary control parameters.
C
   CALL AGSTUP (XYCD(1),1,0,113,2,XYCD(2),1,0,113,2)
C
C Draw the background.
C
   CALL AGBACK
C
EXAMPLES - 136 -
C Draw the curve twice to make it darker.
C
CALL AGCURV (XYCD(1),2,XYCD(2),2,113,1)
CALL AGCURV (XYCD(1),2,XYCD(2),2,113,1)

C Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
C Advance the frame.
C
CALL FRAME
C
STOP
C
Format.
C
1001 FORMAT (14E5.0)
C
END
SUBROUTINE BNDARY
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
6. MESSAGES

AUTOGRAPH routines detect certain errors and, in response, call the routine SETER, which is an adapted version of a PORT error handler. Currently, all such errors are treated as being fatal and cause termination of the job. An error message is logged before the job is terminated. Each such message includes the name of the routine which detected the error and may be accompanied by supplementary information aimed at allowing the user to easily identify the call that caused the error. The possible error messages are as follows (in alphabetical order):

AGEXAX (CALLED BY AGSTUP) - USER-SYSTEM-TO-LABEL-SYSTEM MAPPING IS NOT MONOTONIC

This probably means that you have replaced the default routine AGUTOL with a version of your own, and you've blown it.

AGGETC - PARAMETER TO GET IS NOT INTRINSICALLY OF TYPE CHARACTER

The argument TPGN specifies a parameter which is not intrinsically of type character. See the description of AGGETC, in the section "ROUTINES".

AGGETP OR AGSETP - ATTEMPT TO ACCESS LABEL ATTRIBUTES BEFORE SETTING LABEL NAME

The parameter 'LABEL/NAME.' must be set prior to the call which gave the error message, specifying which label's attributes are being referenced.

AGGETP OR AGSETP - ATTEMPT TO ACCESS LINE ATTRIBUTES BEFORE SETTING LINE NUMBER

The parameter 'LINE/NUMBER.' must be set prior to the call which gave the error message, specifying which label line's attributes are being referenced.

AGGETP OR AGSETP - ILLEGAL KEYWORD USED IN PARAMETER IDENTIFIER

The argument TPGN contains an unrecognizable keyword.

AGKURV - NUMBER OF POINTS IS .LE. 0

The argument NEXY, in a call to AGCURV, is less than or equal to zero. The
routine AGKURV is called by AGCURV to draw un-windowed curves.

AGNUMB - MANTISSA TOO LONG
AGNUMB - EXPONENT TOO LARGE
AGNUMB - ZERO-LENGTH MANTISSA

AGNUMB is called by AGAXIS to generate a character string expressing the value of a floating-point number. You should not be able to generate any of AGNUMB's error messages. If you do, see the AUTOGRAPH specialist.

AGQURV - NUMBER OF POINTS IS .LE. 0

The argument NEXY, in a call to AGCURV, is less than or equal to zero. The routine AGQURV is called by AGCURV to draw windowed curves.

AGRSTR - ERROR ON READ
AGRSTR - END-OF-FILE ON READ

Probably the unit specified by IFNO was not positioned properly.

AGSAVE - ERROR ON WRITE

A system error has occurred as a result of the attempted "WRITE".

AGSETC - PARAMETER TO SET IS NOT INTRINSICALLY OF TYPE CHARACTER

This means that the argument TPGN specifies some parameter other than one of the acceptable possibilities. See the description of AGSETC in the section "ROUTINES".

AGSETP - ATTEMPT TO DEFINE LINE OF NON-EXISTENT LABEL

The user has attempted to define a line of a label without first specifying which label; 'LABEL/NAME.' must be set prior to the call which gave the error message.

AGSETP - LABEL LIST OVERFLOW - SEE AUTOGRAPH SPECIALIST

The user has attempted to define more labels than AUTOGRAPH can handle; a
modification of AUTOGRAF is required. See the paragraph "INFORMATIONAL LABELS", in the section "OVERVIEW", or talk to the AUTOGRAF specialist.

AGSETP - LINE LIST OVERFLOW - SEE AUTOGRAF SPECIALIST

The user has attempted to define more label lines than AUTOGRAF can handle; a modification of AUTOGRAF is required. See the paragraph "INFORMATIONAL LABELS", in the section "OVERVIEW", or talk to the AUTOGRAF specialist.

AGSTCH - CHARACTER-STRING BUFFER OVERFLOW - SEE CONSULTANT

The routine AGSTCH is called by AGSETC to stash the character string in AUTOGRAF's character storage space. The available storage space has been exhausted. See the consultant.

AGSTCH - CHARACTER-STRING INDEX OVERFLOW - SEE CONSULTANT

The routine AGSTCH is called by AGSETC to stash the character string in AUTOGRAF's character storage space. Too many such strings have been stored. See the consultant.

AGSTUP - GRAPH WINDOW IMPROPERLY SPECIFIED

The parameters in the group named 'GRAPH.' have improper values.

AGSTUP - GRID WINDOW IMPROPERLY SPECIFIED

The parameters in the group named 'GRID.' have improper values. This is most likely to occur when 'SET.' has the value "2." or "4.", specifying that the edges of the grid window are to be as implied by the last call to the plot package routine SET. Check to make sure that the portion of the plotter frame specified by the last SET call is within the current graph window.

AGSTUP - s LABELS IMPROPERLY SPECIFIED

where "s" = "LEFT", "RIGHT", "BOTTOM", "TOP", or "INTERIOR".

Re-read the paragraph "THE LABEL BOXES", in the section "OVERVIEW". You have defined a label with a basepoint on one edge of the grid window and an offset vector pointing outward, some part of which extends inside the grid window (or vice-versa). This is not allowed.

MESSAGES - 140 - AUTOGRAF
SUBROUTINE CONRAN (XD,YD,ZD,NDP,WK,IWK,SCRARR)

SUBROUTINE CONRAN(XD,YD,ZD,NDP,WK,IWK,SCRARR)
Standard and smooth versions 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 July 1984

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

ON INPUT XD Array of dimension NDP containing the X-coordinates of the data points.

April, 1986 1 CONRAN
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).

SCRARR  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 CONRAN.

ENTRY POINTS  CONRAN, CONDET, CONINT, CONCAL, CONLOC, CONTNG, CONDRW, CONCLS, CONSTP, CONBDN, CONTLK, CONPDV, CONOP1, CONOP2, CONOP3, CONOP4, CONXCH, CONREO, CONCOM, CONCLD, CONPMM, CONGEN, CONLOD, CONECD, CONOUT, CONOT2, CONSLD, CONLCH, CONLIN, CONDSD, CONSSD

COMMON BLOCKS  CONRA1, CONRA2, CONRA3, CONRA4, CONRA5, CONRA6, CONRA7, CONRA8, CONRA9, CONR0, CONR1, CONR2, CONR3, CONR4, CONR5, CONR6, CONR7, CONR8, CONR9, CONR10, CONR11, CONR12, CONR13, CONR14, CONR15, CONR16, CONR17, RANINT, INTFR 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.

**LANGUAGE**

FORTRAN77

**HISTORY**

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

April, 1986
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 assumes the unsealed data values. See the SDC flag, below.

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

\textbf{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: \texttt{CALL CONOP3('CIL=OFF',0.,0.)}
If user set: \texttt{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: \texttt{CALL CONOP3('CIL=ON',15.,1)}
where 15. represents the contour increment desired by the user.

\textbf{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 are 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: \texttt{CALL CONOP3('CON=OFF',0.,0.)}
If user set: \texttt{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 breakpoint (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

April, 1986 6 CONRAN
default value which is zero.

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

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' of '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. For an explanation of the option value settings see
the OPTN routine in the NCAR system plot package documentation. Briefly, 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—Cl surfaces and linear. The Cl 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 Cl surface.

For Cl surface CALL CONOP1('ITP=Cl')
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 ILMACH.

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 are 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
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 CONOPI('PER=ON')
To turn off: CALL CONOPI('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 CONOPI('PMM=OFF')
To turn on: CALL CONOPI('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.

Draw the shield: CALL CONOPI('PSL=ON')
Don't draw it: CALL CONOPI('PSL=OFF')

REP Flag indicating the use of the same data in a new execution. The default value is OFF.

To turn on: CALL CONOPI('REP=ON')
To turn off: CALL CONOPI('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 CONOPI('SCA=OFF')
Program scaling: CALL CONOPI('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 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 coords. 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.,  
1,7.,7.,10.,10.,7.:/  
CALL CONOP3 (6HSLD=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 WSTR.

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 WSTR.

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.
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 April, 1986
with splines under tension. To adjust the
amount of smoothing applied, adjust the ten-
sion 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 con-
touring 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 ISTRING 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 trian-
gles 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 CONOPl('TRI=ON')

To turn off: CALL CONOPl('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.

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</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>PER</td>
<td>ON</td>
</tr>
<tr>
<td>PMV</td>
<td>OFF</td>
</tr>
<tr>
<td>REP</td>
<td>OFF</td>
</tr>
<tr>
<td>SDC</td>
<td>OFF</td>
</tr>
<tr>
<td>SDL</td>
<td>OFF</td>
</tr>
<tr>
<td>SML</td>
<td>OFF</td>
</tr>
<tr>
<td>SPD</td>
<td>OFF</td>
</tr>
<tr>
<td>SPT</td>
<td>OFF</td>
</tr>
<tr>
<td>STL</td>
<td>OFF</td>
</tr>
<tr>
<td>TEN</td>
<td>OFF</td>
</tr>
<tr>
<td>TFR</td>
<td>ON</td>
</tr>
<tr>
<td>TOP</td>
<td>OFF</td>
</tr>
</tbody>
</table>

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>
<tr>
<td>OPTIONS WHICH EFFECT THE CONTOURS</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--</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>
<tr>
<td>ISZSML</td>
<td>15 plotter address units</td>
</tr>
<tr>
<td>ISZSPD</td>
<td>8 plotter address units</td>
</tr>
<tr>
<td>ISZSTL</td>
<td>16 plotter address units</td>
</tr>
<tr>
<td>ISTEP</td>
<td>40</td>
</tr>
<tr>
<td>IVAL</td>
<td>'HI' for all except minor contour lines which are 'LO'.</td>
</tr>
<tr>
<td>L</td>
<td>7 characters (including both parentheses)</td>
</tr>
<tr>
<td>NCL</td>
<td>Computed by the program based on input data. Up to 30 contour levels are permitted.</td>
</tr>
<tr>
<td>NUM</td>
<td>4 data points</td>
</tr>
<tr>
<td>SCALE</td>
<td>1. (no scaling performed)</td>
</tr>
<tr>
<td>TENS</td>
<td>2.5</td>
</tr>
<tr>
<td>ICSD</td>
<td>0 (no shield)</td>
</tr>
</tbody>
</table>

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 IPCO 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 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

Akima, Hirosha
A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points.
ACM Transactions on Mathematical Software vol 4, no. 2, June 1978, pages 148-159

Lawson, C.L.
Software for C1 Surface Interpolation
JPL Publication 77-30
August 15, 1977

April, 1986
<table>
<thead>
<tr>
<th>CONRAN ERROR MESSAGES</th>
<th>ERROR ROUTINE</th>
<th>MESSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONRAN</td>
<td>INPUT PARAMETER NDP LT NCP</td>
</tr>
<tr>
<td>2</td>
<td>CONRAN</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(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.

Note: The common blocks listed include all the common used by the entire CONRAN family. Not all members will use all the common variables.

CONRAL
- CL-array of contour levels
- NCL-number of contour levels
- OLDZ-Z value of left neighbor to current location
- PV-array of previous row values
- HI-largest contour plotted
- FLO-lowest contour plotted
- FINC-increment level between equally spaced contours

CONRA2

April, 1986 19 CONRAN
REPEAT-flag to triangulate and draw or just draw
EXTRAP-plot data outside of convex data hull
PER-put perimeter around plot
MESS-flag to indicate message output
ISCALE-scaling switch
LOOK-plot triangles flag
PLDVL-S-plot the data values flag
GRD-plot grid flag
CON-user set or program set contours flag
CINC-user or program set increment flag
CHILO-user or program set hi low contours
LABON-flag to control labeling of contours
PMIMX-flag to control the plotting of min's
and max's
SCALE-the scale factor for contour line values
and min, max plotted values
FRADV-advance frame before plotting triangulation
EXTRI-only plot triangulation
BPSIZ-breakpoint size for dashpatterns
LISTOP-list options on UNIT6 flag

CONRA3
IRED-ERPRT77 recoverable error flag

CONRA4
NCP-number of data points used at each point for
  polynomial construction.
NCPSZ-max size allowed for NCP

CONRA5
NIT-flag to indicate status of search data base
ITIPV-last triangle interpolation occurred in

CONRA6
XST-X coordinate start point for contouring
YST-Y coordinate start point for contouring
XED-X coordinate end point for contouring
YED-Y coordinate end point for contouring
STPSZ-step size for X,Y change when contouring
IGRAD-number of graduations for contouring (step size)
IG-reset value for IGRAD
XRG-X range of coordinates
YRG-Y range of coordinates
BORD-percent of frame used for contour plot
PXST-X plotter start address for contours
PYST-Y plotter start address for contours
PXED-X plotter end address for contours
PYED-Y plotter end address for contours
ITICK-number of tick marks for grids and perimeters

CONRA7
TITLE-switch to indicate if title option ON or OFF
ISTRNG-character string containing the title
ITCNT-character count of ISTRNG
ITLTSZ-size of title in PWRT units

CONRA8

April, 1986 20

CONRAN
IHIGH-default intensity setting
INMAJ-contour Level intensity for major lines
INMIN-contour Level intensity for minor lines
INLAB-title and message intensity
INDAT-data value intensity
FORM-the format for plotting the data values
LEN-the number of characters in the format
IFMT-size of the format field
LEND-default format length
IFMTD-default format field size
ISIZEP-size of the plotted data values

CONRA9
X-array of X coordinates of contours drawn at current contour level
Y-array of Y coordinates of contours drawn at current contour level
NP-count in X and Y
MXXX-size of X and Y
TR-top right corner value of current cell
BR-bottom right corner value of current cell
TL-top left corner value of current cell
BL-bottom left corner value of current cell
CONV-current contour value
XN-X position where contour is being drawn
YN-Y position where contour is being drawn
ITLL-triangle where top left corner of current cell lies
IBLL-triangle of bottom left corner
ITRL-triangle of top right corner
IBRL-triangle of bottom right corner
XC-X coordinate of current cell
YC-Y coordinate of current cell
ITLOC-in conjunction with PV stores the triangle where PV value came from

CONR10
NT-number of triangles generated
NL-number of line segments
NTNL-NT+NL
JWIPT-pointer into IWK where where triangle point numbers are stored
JWIWL-in IWK the location of a scratch space
JWIWP-in IWK the location of a scratch space
JWIPL-in IWK the location of end points for border line segments
IPR-in WK the location of the partial derivatives at each data point
ITPV-the triangle where the previous value came from

CONR11
NREP-number of repetitions of dash pattern before a label
NCRT-number of CRT units for a dash mark or blank
ISIZEL-size of contour line labels
NDASH-array containing the negative valued contour dash
pattern
MINGAP-number of unlabeled lines between each labeled one
IDASH-positive valued contour dash pattern
ISIZEM-size of plotted minimums and maximums
EDASH-equal valued contour dash pattern
TENS-default tension setting for smoothing

CONR12
IXMAX, IYMAX-maximum X and Y coordinates relative to the
    scratch array, SCRARR
XMAX, YMAX-maximum X and Y coordinates relative to users
    coordinate space

CONR13
XVS-array of the X coordinates for shielding
YVS-array of the Y coordinates for shielding
IXVST-pointer to the users X array for shielding
IYVST-pointer to the users Y array for shielding
ICOUNT-count of the shield elements
SPVAL-special value used to halt contouring at the shield
    boundary
SHIELD-logical flag to signal status of shielding
SLDPLT-logical flag to indicate status of shield plotting

CONR14
LINEAR-C1 linear interpolating flag

CONR15
ISTRNG-title of the plot

CONR16
FORM-Format used for data

CONR17
NDASH-Dash pattern used for contour lines less than BP
IDASH-Dash pattern used for contour lines greater than BP
EDASH-Dash pattern used for contour lines equal to the BP

RANINT
IRANMJ-color index for normal (major) intensity lines
IRANMN-color index for low intensity lines
IRANMJ-color index for text (labels)
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
July 1984

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: CL surfaces and linear.

USAGE
CALL CONRAQ(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 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.
Array of dimension NDP containing the Y-coordinates of the data points.

Array of dimension NDP containing the data values at the points.

Number of data points (must be 4 or greater) to be contoured.

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)*NDP). When using linear interpolation the array must be at least IWK((27+4)*NDP).

All arguments remain unchanged except the scratch arrays IWK and WK, which have written into. If making multiple runs on the same triangulation, IWK and WK must be saved and returned to the next invocation of CONRAQ.

CONRAQ, CONDET, CONINT, CONCAL, CONLOC, CONTNG, CONDRW, CONCLS, CONSTP, CONBDN, CONTLK, CONTOR, CONPDV, CONOP1, CONOP2, CONOP3, CONOP4, CONXCH, CONOUT, CONOT2, CONLIN

CONRA1, CONRA2, CONRA3, CONRA4, CONRA5, CONRA6, CONRA7, CONRA8, CONRA9, CONRA10, CONRA11, CONRA12, CONRA14, CONRA15, CONRA16, CONRA17, RAQINT

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.

Single

GRIDAL, CONCOM, CONTERP, the ERPRT77 package, and the SPPS.

April, 1986
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

HISTORY

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

April, 1986
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 par-
tial derivatives at each data point.
The estimated partial derivatives are
used for the construction of the interpolat-
ing 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 in-
put 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 incre-
ment and contour high and low values assumes
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 are 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.

DEF
Reset flags to default values. Activating this option sets all flags to the default value. DEF has no 'ON' of '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: 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 con-
tour 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. For an
explanation of the option value settings see
the OPTN routine in the NCAR system plot
package documentation. Briefly, IVAL values
range from 0 to 255 or the character strings
'LO and 'HI'. The default is 'HI' except

April, 1986
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 de-
fault 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 execu-
tion time in seconds. If PER or GRI are 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.

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 WISTR.

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 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.

CHL=OFF LOT=OFF SCA=ON TOP=OFF
CIL=OFF MES=ON SPD=OFF TRI=OFF
CON=OFF NCP=OFF SSZ=OFF ITP=C1
EXT=OFF PDV=OFF STL=OFF

April, 1986

CONRAQ
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>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>CHAR</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>ISZSPD</td>
<td>8 plotter address units</td>
</tr>
<tr>
<td>ISZSTL</td>
<td>16 plotter address units</td>
</tr>
<tr>
<td>ISTEP</td>
<td>40</td>
</tr>
<tr>
<td>IVAL</td>
<td>'HI' for all except minor contour lines which are 'LO'.</td>
</tr>
<tr>
<td>L</td>
<td>7 characters (including both parentheses)</td>
</tr>
<tr>
<td>NCL</td>
<td>Computed by the program based on input data. Up to 30 contour levels are permitted.</td>
</tr>
<tr>
<td>NUM</td>
<td>4 data points</td>
</tr>
</tbody>
</table>
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 > 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.

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

**REFERENCES**

Akima, Hirosha

A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points.
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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.

Note: The common blocks listed include all the common used by the entire CONRAN family. Not all members will use all the common variables.

CONRAQ
CL-array of contour levels
NCL-number of contour levels

April, 1986 15 CONRAQ
OLDZ-Z value of left neighbor to current location
PV-array of previous row values
HI-largest contour plotted
FLO-lowest contour plotted
FINC-increment level between equally spaced contours

CONRA2
REPEAT-flag to triangulate and draw or just draw
EXTRAP-plot data outside of convex data hull
PER-put perimeter around plot
MESS-flag to indicate message output
ISCALE-scaling switch
LOOK-plot triangles flag
PLDVL3-plot the data values flag
GRD-plot grid flag
CON-user set or program set contours flag
CINC-user or program set increment flag
CHILO-user or program set hi low contours
LABON-flag to control labeling of contours
PMIMX-flag to control the plotting of min's and max's
SCALE-the scale factor for contour line values and min, max plotted values
FRADV-advance frame before plotting triangulation
EXTRI-only plot triangulation
BPSIZ-breakpoint size for dashpatterns
LISTOP-list options on UNIT6 flag

CONRA3
IRED-ERPRT77 recoverable error flag

CONRA4
NCP-number of data points used at each point for polynomial construction.
NCPN-1-max size allowed for NCP

CONRA5
NIT-flag to indicate status of search data base
ITIPV-last triangle interpolation occurred in

CONRA6
XST-X coordinate start point for contouring
YST-Y coordinate start point for contouring
XED-X coordinate end point for contouring
YED-Y coordinate end point for contouring
STPSZ-step size for X,Y change when contouring
IGRAD-number of graduations for contouring (step size)
IG-reset value for IGRAD
XRG-X range of coordinates
YRG-Y range of coordinates
BORD-percent of frame used for contour plot
PXST-X plotter start address for contours
PYST-Y plotter start address for contours
PXED-X plotter end address for contours
PYED-Y plotter end address for contours
ITICK-number of tick marks for grids and perimeters
CONRA7

TITLE-switch to indicate if title option ON or OFF
ISTRING-character string containing the title
ICNT-character count of ISTRING
ITLSIZ-size of title in PWRIT units

CONRA8

IHIGH-default intensity setting
INMAJ-contour Level intensity for major lines
INMIN-contour Level intensity for minor lines
INLAB-title and message intensity
INDAT-data value intensity
FORM-the format for plotting the data values
LEN-the number of characters in the format
IFMT-size of the format field
LEND-default format length
IFMTD-default format field size
ISIZEP-size of the plotted data values

CONRA9

X-array of X coordinates of contours drawn at current contour level
Y-array of Y coordinates of contours drawn at current contour level
NP-count in X and Y
MXXY-size of X and Y
TR-top right corner value of current cell
BR-bottom right corner value of current cell
TL-top left corner value of current cell
BL-bottom left corner value of current cell
CONV-current contour value
XN-X position where contour is being drawn
YN-Y position where contour is being drawn
ITLL-triangle where top left corner of current cell lies
IBLL-triangle of bottom left corner
ITRL-triangle of top right corner
IBRL-triangle of bottom right corner
XC-X coordinate of current cell
YC-Y coordinate of current cell
ITLOC-in conjunction with PV stores the triangle where PV value came from

CONRA10

NT-number of triangles generated
NL-number of line segments
NTNL-NT+NL
JWIPT-pointer into IWK where where triangle point numbers are stored
JWIWL-in IWK the location of a scratch space
JWIPW-in IWK the location of a scratch space
JWIPL-in IWK the location of end points for border line segments
IPR-in WK the location of the partial derivatives at each data point

April, 1986
ITPV-the triangle where the previous value came from

CONR11
NREP-number of repetitions of dash pattern before a label
NCRT-number of CRT units for a dash mark or blank
ISIZEL-size of contour line labels
NDASH-array containing the negative valued contour dash pattern
MINGAP-number of unlabeled lines between each labeled one
IDASH-positive valued contour dash pattern
ISIZEM-size of plotted minimums and maximums
EDASH-equal valued contour dash pattern
TENS-default tension setting for smoothing

CONR12
IXMAX, IYMAX-maximum X and Y coordinates relative to the scratch array, SCRARR
XMAX, YMAX-maximum X and Y coordinates relative to users coordinate space

CONR13
XVS-array of the X coordinates for shielding
YVS-array of the Y coordinates for shielding
IXVST-pointer to the users X array for shielding
IYVST-pointer to the users Y array for shielding
ICOUNT-count of the shield elements
SPVAL-special value used to halt contouring at the shield boundary
SHIELD-logical flag to signal status of shielding
SLDPLT-logical flag to indicate status of shield plotting

CONR14
LINEAR-C1 linear interpolating flag

CONR15
ISTRNG-title of the plot

CONR16
FORM-Format used for data

CONR17
NDASH-Dash pattern used for contour lines less than BP
IDASH-Dash pattern used for contour lines greater than BP
EDASH-Dash pattern used for contour lines equal to the BP

RAQINT
IRAQMJ-color index for normal (major) intensity lines
IRAQMN-color index for low intensity lines
IRAQMJ-color index for text (labels)
CONRAS
SUBROUTINE CONRAS (XD,YD,ZD,NDP,WK,IWK,SCRARR)

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 July 1984

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

April, 1986 1 CONRAS
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 Cl 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).

SCRARR
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, CONDEW, CONCLS, CONSTP, CONBDN, CONTLK, CONPDV, CONOP1, CONOP2, CONOP3, CONOP4, CONXCH, CONREQ, CONCOM, CONCLD, CONPMS, CONSTP, CONGEN, CONLOD, CONECD, CONOUT, CONOT2, CONCLD, CONSLD, CONLCM, CONLIN, CONDSD, CONSSD

COMMON BLOCKS
CONRA1, CONRA2, CONRA3, CONRA4, CONRA5, CONRA6, CONRA7, CONRA8, CONRA9, CONR10, CONR11, CONR12, CONR13, CONR14, CONR15, CONR16, CONR17, RASINT, INTPR from the DASH package

April, 1986
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.

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

HISTORY

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 Hiroshi 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 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 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 assumes 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 are 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)

April, 1986

CONRAS5
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' of '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: 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. For an explanation of the option value settings see the OPTN routine in the NCAR system plot package documentation. Briefly, 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: 

```fortran
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 

```fortran
CALL CONOP1('ITP=C1')
```

For linear 

```fortran
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: 

```fortran
CALL CONOP1('LAB=ON')
```

To turn off: 

```fortran
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: 

```fortran
CALL CONOP1('LOT=ON')
```

To turn off: 

```fortran
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.

April, 1986

9
CONRAS
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 are 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 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 entire array, the number of (X + Y) shield coords. 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 (6HSLD=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 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 CONOPl('TFR=ON')

To turn off: CALL CONOPl('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 CONOPl('TOP=OFF')

To turn on: CALL CONOPl('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.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>DEFAULT VALUES</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>SLD</td>
<td>OFF</td>
</tr>
<tr>
<td>SDC</td>
<td>OFF</td>
</tr>
<tr>
<td>SML</td>
<td>OFF</td>
</tr>
<tr>
<td>SPD</td>
<td>OFF</td>
</tr>
<tr>
<td>STL</td>
<td>OFF</td>
</tr>
<tr>
<td>TEN</td>
<td>OFF</td>
</tr>
<tr>
<td>TFR</td>
<td>ON</td>
</tr>
<tr>
<td>TOP</td>
<td>OFF</td>
</tr>
<tr>
<td>TRI</td>
<td>OFF</td>
</tr>
</tbody>
</table>

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
- DAS=OFF
- DBP=OFF
- EXT=OFF
- FMT=OFF
- GRI=OFF
- ITP=C1
- LAB=ON
- LOT=OFF
- LSZ=OFF
- MES=ON
- SLD=OFF
- SDC=OFF
- SML=OFF
- SPD=OFF
- STL=OFF
- TEN=OFF
- TFR=ON
- TOP=OFF
- TRI=OFF

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</td>
</tr>
</tbody>
</table>
range of HI and LO values of the input data.

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

The shape of the contours may be modified by changing NCP and SSZ. NCP controls the

April, 1986 17 CONRAS
CONTOURS 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 DSQ0 and IPCO 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 Cl 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 Akima, Hirosha
A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points.
ACM Transactions on Mathematical Software vol 4, no. 2, June 1978, pages 148-159
Lawson, C.L.
Software for Cl Surface Interpolation

April, 1986 18 CONRAS
## 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>
<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.

Note: The common blocks listed include all the common used by the entire CONRAN family. Not all members will use all the common variables.

**CONRA1**
- CL-array of contour levels
- NCL-number of contour levels
- OLDZ-Z value of left neighbor to current location
- PV-array of previous row values
- HI-largest contour plotted
- FLO-lowest contour plotted
FINC—increment level between equally spaced contours

CONRA2
REPEAT—flag to triangulate and draw or just draw
EXTRAP—plot data outside of convex data hull
PER—put perimeter around plot
MESS—flag to indicate message output
ISCALE—scaling switch
LOOK—plot triangles flag
PLDVLS—plot the data values flag
GRD—plot grid flag
CON—user set or program set contours flag
CINC—user or program set increment flag
CHILO—user or program set hi low contours
LABON—flag to control labeling of contours
PMIMX—flag to indicate the plotting of min's
and max's
SCALE—the scale factor for contour line values
and min, max plotted values
FRADV—advance frame before plotting triangulation
EXTRI—only plot triangulation
BPSIZ—breakpoint size for dashpatterns
LISTOP—list options on UNIT6 flag

CONRA3
IRED—ERPRT77 recoverable error flag

CONRA4
NCP—number of data points used at each point for
polynomial construction.
NCPSZ—max size allowed for NCP

CONRA5
NIT—flag to indicate status of search data base
ITIPV—last triangle interpolation occurred in

CONRA6
XST—X coordinate start point for contouring
YST—Y coordinate start point for contouring
XED—X coordinate end point for contouring
YED—Y coordinate end point for contouring
STPSZ—step size for X,Y change when contouring
IGRAD—number of graduations for contouring (step size)
IG—reset value for IGRAD
XRG—X range of coordinates
YRG—Y range of coordinates
BORD—percent of frame used for contour plot
PXST—X plotter start address for contours
PYST—Y plotter start address for contours
PXED—X plotter end address for contours
PYED—Y plotter end address for contours
ITICK—number of tick marks for grids and perimeters

CONRA7
TITLE—switch to indicate if title option ON or OFF
ISTRNG—character string containing the title
ICNT—character count of ISTRNG

April, 1986  20  CONRAS
ITLSIZ—size of title in PWRIT units

CONRA
IHIGH—default intensity setting
INMAJ—contour Level intensity for major lines
INMIN—contour Level intensity for minor lines
INLAB—title and message intensity
INDAT—data value intensity
FORM—the format for plotting the data values
LEN—the number of characters in the format
IFMT—size of the format field
LEND—default format length
IFMTD—default format field size
ISIZEP—size of the plotted data values

CONRA9
X—array of X coordinates of contours drawn at current contour level
Y—array of Y coordinates of contours drawn at current contour level
NP—count in X and Y
MXXY—size of X and Y
TR—top right corner value of current cell
BR—bottom right corner value of current cell
TL—top left corner value of current cell
BL—bottom left corner value of current cell
CONV—current contour value
XN—X position where contour is being drawn
YN—Y position where contour is being drawn
ITLL—triangle where top left corner of current cell lies
IBLL—triangle of bottom left corner
ITRL—triangle of top right corner
IBRL—triangle of bottom right corner
XC—X coordinate of current cell
YC—Y coordinate of current cell
ITLOC—in conjunction with PV stores the triangle where PV value came from

CONR10
NT—number of triangles generated
NL—number of line segments
NTNL—NT+NL
JWIPT—pointer into IWK where where triangle point numbers are stored
JWIWL—in IWK the location of a scratch space
JWIWP—in IWK the location of a scratch space
JWIPL—in IWK the location of end points for border line segments
IPR—in WK the location of the partial derivatives at each data point
ITPV—the triangle where the previous value came from

CONR11
NREP—number of repetitions of dash pattern before a label
NCRT—number of CRT units for a dash mark or blank
ISIZEL—size of contour line labels
NDASH—array containing the negative valued contour dash pattern
MINGAP—number of unlabeled lines between each labeled one
IDASH—positive valued contour dash pattern
ISTZEM—size of plotted minimums and maximums
EDASH—equal valued contour dash pattern
TENS—default tension setting for smoothing

CONR12
IXMAX, IYMAX—maximum X and Y coordinates relative to the scratch array, SCRARR
XMAX, YMAX—maximum X and Y coordinates relative to users coordinate space

CONR13
XVS—array of the X coordinates for shielding
YVS—array of the Y coordinates for shielding
IXVST—pointer to the users X array for shielding
IYVST—pointer to the users Y array for shielding
ICOUNT—count of the shield elements
SPVAL—special value used to halt contouring at the shield boundary
SHIELD—logical flag to signal status of shielding
SLDPLT—logical flag to indicate status of shield plotting

CONR14
LINEAR—Cl linear interpolating flag

CONR15
ISTRING—title of the plot

CONR16
FORM—Format used for data

CONR17
NDASH—Dash pattern used for contour lines less than BP
IDASH—Dash pattern used for contour lines greater than BP
EDASH—Dash pattern used for contour lines equal to the BP

RASINT
IRASMJ—color index for normal (major) intensity lines
IRASMN—color index for low intensity lines
IRASMJ—color index for text (labels)
CONREC
SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

DIMENSION OF ARGUMENTS
Z(L,N)

LATEST REVISION
June 1984

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 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
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
The (origin of the) array to be contoured. Z is dimensioned L by N.

April, 1986
1

CONREC
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
\( \geq 0 \) Increment between contour levels.
\( = 0 \) A value, which produces between 10 and 30 contour levels at nice values, is generated by CONREC.
\( \leq 0 \) The number of levels generated by CONREC is \( \text{ABS}(\text{FINC}) \).

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, STLNE, DRLINE, MINMAX, PNTVAL, CALCNT, EZCNTR, CONBD

COMMON BLOCKS
INTPR, RECEINT, CONRE1, CONRE2, CONRE3, CONRE4, CONRE5

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 GRIDAL, the ERPRT77 package, and the SPPS.

April, 1986 3 CONREC
Plots contour map.

Single

FORTRAN 77

Replaces old contouring package called CALCNT at NCAR.

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.

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.

Varies widely with size and smoothness of Z.

<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>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>
</tbody>
</table>
| NLM    | 40      | Maximum number of contour levels. If this is to be

April, 1986
increased, the dimensions of CL and RWORK in CONREC must be increased by the same amount.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</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</td>
</tr>
</tbody>
</table>

April, 1986 5 CONREC
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.

ISOLID 1023 Dash pattern for non-negative contour lines.
CONRECSUPR
SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

DIMENSION OF
ARGUMENTS
Z(L,N)

LATEST REVISION
June 1984

PURPOSE
CONRECSUPR 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.

April, 1986

1

CONRCSPR
ON INPUT Z
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 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
All arguments are unchanged.
FOR CONREC

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.

NOTE FOR NCAR USERS
This routine is NOT part of the default

April, 1986

3

CONRCSPR
libraries at NCAR. CONRECSUPR must be acquired, compiled and loaded to be used at NCAR.

Plots contour map.

Single

FORTRAN 77

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.

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.

Varies widely with size and smoothness of Z.

<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>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>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------</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</td>
</tr>
</tbody>
</table>
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.

**ISOLID** 1023 Dash pattern for non-negative contour lines.

**IHILO** 3 Flag to control labelling of highs, lows, or both: if NHIL = 0, then
  - IHILO = 0 means do not label highs nor lows
  - IHILO = 1 means highs are labelled, lows are not
  - IHILO = 2 means lows are

April, 1986

CONRCSPR
labelled, highs are not
. IHILO = 3 means both
highs and lows are labelled
CONRECQCK
SUBROUTINE CONREC (Z,L,M,N,FLO,HI,FINC,NSET,NHI,NDOT)

DIMENSION OF ARGUMENTS
Z(L,N)

LATEST REVISION
June 1984

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.

April, 1986
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, CALCINT, E2CNTR, CONBD

COMMON BLOCKS
CONREL, CONRE4, CONRE5

REQUIRED LIBRARY Routines
GRIDAL, the ERPRT77 package, and the SPPS. DASHCHAR is also needed.

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.

April, 1986
A faster version of CONREC without line labelling capabilities.

The grid space is divided into \( L \times N \) 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.

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.

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>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>
</tbody>
</table>

April, 1986
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLT</td>
<td>0.05</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>0.05</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>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>0.25</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>SPVAL</td>
<td>0.0</td>
<td>Contains the special value</td>
</tr>
</tbody>
</table>

April, 1986

CONRCQCK
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.
SUBROUTINE DASHDC (IPAT,JCRT,JSIZE)

Software dashed line package with character capability

LATEST REVISION

May 1984

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), 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

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.

April, 1986

DASHCHAR
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.

ARGUMENTS 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.

NOTE
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.

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.

April, 1986
Plots solid or dashed lines, possibly with characters at intervals in the line.

**PRECISION**

Single

**LANGUAGE**

FORTRAN

**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.

**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**

FORTRAN77

**INTERNAL PARAMETERS**

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEFAULT</th>
<th>FUNCTION</th>
<th>DECLARED IN BLOCK DATA</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,</td>
<td></td>
</tr>
</tbody>
</table>

April, 1986

3

DASHCHAR
which reduces the probability of the label being written on top of a label of a nearby line drawn with FPART = 1.

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).
SUBROUTINE DASHDB (IPAT)

Software dashed line package

LATEST REVISION April 1984

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)

ARGUMENTS
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)

ARGUMENTS 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.

April, 1986

DASHLINE
ENTRY POINTS
DASHDB, CURVED, FRSTD, VECTD, LINED, LASTD, CFVLD, DRAWPV, DASHBD

COMMON BLOCKS
INTRPR, DSHD, DSHDA, DSHDD, DSHDC, DSHDB

REQUIRED LIBRARY ROUTINES
The ERPRT77 package and the SPPS.

PRECISION
Single

LANGUAGE
FORTRAN

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
FORTRAN77

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</td>
</tr>
</tbody>
</table>

April, 1986
current pen position (distance=
difference in X-coordinates +
difference in Y-coordinates).
DASHSMTH
SUBROUTINE FDVDLD (IENTRY,IIX,IIY)

Software dashed line package with character capability and smoothing

LATEST REVISION

June 1984

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) 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.

ARGUMENTS 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.

NOTE
When using FRSTD and VECTD, LASTD must be called (no arguments needed). LASTD sets up the calls to the smoothing routines KURVL1S and KURV2S.

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, KURVL1S, KURV2S, CFVLD, FDVLD, DRAWPV, DASHBD

COMMON BLOCKS
INTPR, DASHD1, DASHD2, DDFLAG, DCFLAG, DSAVEL, DSAVE2, DSAVE3, DSAVE5, CFFLAG, SMFLAG, DFFLAG, FDFLAG

REQUIRED LIBRARY ROUTINES
The ERPRT77 package and the SPPS.

April, 1986 2 DASHSMTH
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

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.

FDVDLD RECEIVES IN ITS ARGUMENTS THE POINTS TO BE PROCESSED FOR A LINE SEGMENT. IT PASSES THESE POINTS TO THE ROUTINES KURV1S AND KURV2S WHICH COMPUTE SPLINES UNDER TENSION PASSING THROUGH THESE POINTS. FDVDLD THEN CALLS CFVLD TO CONNECT THE POINTS GENERATED IN KURV2S.

April, 1986
DASHSUPR
SUBROUTINE FDVDLD (IENTRY,IIX,IIY)

Software dashed line package with character capability, smoothing, and the capability of removing crowded lines.

LATEST REVISION

June 1984

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) 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.

ARGUMENTS 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).

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 using FRSTD and VECTD, LASTD must be called (no arguments needed). LASTD sets up the calls to the smoothing routines KURV1S and KURV2S.

When switching from the regular plotting

April, 1986
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, CFVL, FDVDLD, DRAWPV, DASHBD, CUTUP, REMOVE, MARKL, PWRTM

COMMON BLOCKS
INTPR, DASHD1, DASHD2, DDFLAG, DCFLAG, DSAVE1, DSAVE2, DSAVE3, DSAVE4, DASVE5, CFFLAG, SMFLAG, DFFLAG

REQUIRED LIBRARY Routines
The ERPRT77 package and the SPPS.

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

HISTORY
Originally written by Tom Reid at Texas A & M. Made portable in 1977 for all machines which support plotters with up to 15 bit resolution. Converted to FORTRAN77 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.

April, 1986
About four times as long as DASHCHAR.

**PORTABILITY FORTRAN77**

**INTERNAL PARAMETERS**

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEFAULT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IPAU</strong></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><strong>FPART</strong></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. |
| **IOFFS** | 0       | Flag to turn on smoothing code.  
- 0 Smoothing.  
- 1 No smoothing. |
| **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. |

April, 1986
SMALL 128. Minimum distance in metacode address units between points which 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.

LL 70 The maximum number of points saved at one time. If there are more than LL points on a given line, LL points are processed, then the next LL, until the entire line is processed. Smoothness between segments is maintained automatically. If LL is increased, the dimensions of XSAVE, YSAVE, XP, YP, and TEMP in FDVDDL must be increased to the new value of LL.

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.
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).

*** IMPLEMENTATION INSTRUCTIONS: 

1.) SET THE MACHINE AND MODEL DEPENDENT CONSTANTS AND DECLARATIONS IN SUBROUTINE REMOVE (SEE COMMENTS IN REMOVE FOR DETAILS).

2.) SET THE CONSTANT ISKIP IN BLOCK DATA DASHBD. SEE COMMENTS IN DASHBD FOR DETAILS.

END OF IMPLEMENTATION INSTRUCTIONS

FDVDLD RECEIVES IN ITS ARGUMENTS THE POINTS TO BE PROCESSED FOR A LINE SEGMENT. IT PASSES THESE POINTS TO THE ROUTINES KURV1S AND KURV2S WHICH COMPUTE SPLINES UNDER TENSION PASSING THROUGH THESE POINTS.

April, 1986
FDVLD THEN CALLS CFVLD TO CONNECT THE POINTS GENERATED IN KURV2S.
EZMAP
EZMAP

May, 1985

David J. Kennison

NCAR, P.O. Box 3000, Boulder, Colorado 80307
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6</td>
<td>EXAMPLE 6.</td>
<td>47</td>
</tr>
<tr>
<td>7.7</td>
<td>EXAMPLE 7.</td>
<td>50</td>
</tr>
<tr>
<td>7.8</td>
<td>EXAMPLE 8.</td>
<td>56</td>
</tr>
<tr>
<td>7.9</td>
<td>EXAMPLE 9.</td>
<td>59</td>
</tr>
<tr>
<td>7.10</td>
<td>A FINAL EXAMPLE.</td>
<td>70</td>
</tr>
</tbody>
</table>
1. 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.

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 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 intensity of various parts of the map are all user-settable parameters. More complete control of intensity, color, 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 MAPES is called by EZMAP once for each outline segment. 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.

A routine called EZMAP, which was implemented in such a way as to cause portability problems, no longer exists (as of March, 1985). Statistics indicated that it was not being used, anyway.
2. PROJECTIONS

EZMAP offers ten different projections, in three groups: conical, azimuthal, and cylindrical.

2.1 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.
2.2 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:

- 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: 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.

- Step 6: Set up linear scales along the u and v axes. The ranges of u and v depend on the projection type.

- 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), \( \sin A \) is the sine of \( A \), \( \cos A \) 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: \( R = \frac{\sin(A/2)}{\cos A} \). 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: \( R = \frac{2 \sin A}{\sqrt{2 \cdot (1 + \cos A)}} \). As \( A \) approaches 180 degrees, \( R \) approaches 2. The entire globe is projected into a circle of radius 2. See example 5.

- Gnomonic: \( R = \frac{\tan A}{\sin A} \). 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: \( R = \sqrt{(S_A^2 - 1) \cdot \sin A / (S_A - 4 \cdot I A)} \), where \( S_A \) 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 \( \cos A < 1 / S_A \) 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.

2.3 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 out-
ward 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

\[
\begin{align*}
U &= RLON \\
V &= RLAT
\end{align*}
\]

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

\[
\begin{align*}
U &= RLON \times \pi/180 \text{ (where } \pi = 3.14159 \ldots) \\
V &= \text{ALOG}(\cot(45 - RLAT/2))
\end{align*}
\]

The entire globe is projected into an infinite rectangle in the u/v plane. U ranges from -\(\pi\) to +\(\pi\), V from -infinity to +infinity. 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

\[
\begin{align*}
U &= RLON/90 \\
V &= \text{COS}(90 - RLAT)
\end{align*}
\]

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.
3. ERROR CONDITIONS

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.0) 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, ENTS, NERROR, EPRINT, and ERFF, 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 MAPPOS - ARGUMENTS ARE INCORRECT
20 MAPRST - ERROR ON READ
21 MAPRST - EOF ON READ
22 MAPSAV - ERROR ON WRITE
```
4. 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

```fortran
PROGRAM CONVRT
   DIMENSION FLIM(4),PNTS(200)
   1  READ (1,3,END=2) NPTS,IGID,(FLIM(I),I=1,4)
      IF (NPTS.GT.1) READ (1,4,END=2) (PNTS(I),I=1,NPTS)
      WRITE (2) NPTS,IGID,(FLIM(I),I=1,4),(PNTS(I),I=1,NPTS)
      GO TO 1
   2  STOP
   3  FORMAT (2I8,4F8.3)
   4  FORMAT (10F8.3)
END
```

with the PORTLIB file EZMAPDAT 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 Dicommed D48 film recorders.

As it stands, the code is a non-GKS version of EZMAP. The code for the GKS version is embedded in it, but commented out. If a GKS version is required, look for all occurrences of the strings 'OLD' and 'GKS' in columns 74-76 and do the obvious: Comment out the "OLD" code and un-comment the GKS code. What you will get is not a "pure" GKS version; it calls routines in NCAR's System Plot Package Simulator (SPPS).
5. MISCELLANY

5.1 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. Cicely Ridley, Jay Chalmers, and Dave Kennison (the current curator) have all had a hand in the creation of this package.

5.2 I/O. Graphical output is generated by calls to the routines FRSTD, VECTD, PWRIT, and POINTS.

Outline data is read from a FORTRAN logical unit. On the Cray, 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.

5.3 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 Dicom 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.

5.4 TIMING. The time required to draw a single map is highly variable and depends on how certain parameters are set.

The March, 1985, update has made EZMAP run significantly slower. This is principally because the default resolution has been increased from a value suitable for the dd80 to a value suitable for the Dicom. 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).

5.5 REFERENCES.


Parker, R.L., "2UCSD SUPERMAP: World Plotting Package".

6. ROUTINES

This section describes, in alphabetical order, all of the user-callable routines in the EZMAP package.

6.1 MAPDRW. Draws the complete map described by the current values of the EZMAP parameters.

6.1.1 MAPDRW - 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.

6.1.2 MAPDRW - ARGUMENTS. None.

6.2 MAPEOS (NOUT, NSEG, IGID, NPTS, PNTS). To examine each outline-dataset segment.

6.2.1 MAPEOS - USAGE. MAPEOS 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.

6.2.2 MAPEOS - ARGUMENTS. INTEGER NOUT, NSEG, IGID, NPTS; REAL PNTS(NPTS).

NOUT is the number of the outline dataset from which the segment comes, as follows:

<table>
<thead>
<tr>
<th>NOUT</th>
<th>Dataset to which segment belongs.</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>

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.

IGID identifies the group to which the segment belongs, as follows:

<table>
<thead>
<tr>
<th>IGID</th>
<th>Group to which segment belongs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continental outlines.</td>
</tr>
<tr>
<td>2</td>
<td>U.S. state boundaries.</td>
</tr>
<tr>
<td>3</td>
<td>International boundaries.</td>
</tr>
</tbody>
</table>
NPTS, on entry, is the number of points defining the outline segment. NPTS may be zeroed by MAPEOS 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.

6.3 MAPFST (RLAT,RLON). Draws lines on the map produced by a call to MAPDRW - used in conjunction with MAPVEC.

6.3.1 MAPFST - USAGE. The statement

CALL MAPFST (RLAT,RLON)

is equivalent to the statement

CALL MAPIT (RLAT,RLON,0)

See the description of MAPIT.

6.3.2 MAPFST - ARGUMENTS. REAL RLAT,RLON.

RLAT and RLON are defined as for MAPIT (which see).

6.4 MAPGRD. Draws a grid.

6.4.1 MAPGRD - 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.

6.4.2 MAPGRD - ARGUMENTS. None.

6.5 MAPGTx (WHICH,XVAL). Gets the current value of a specified EZMAP parameter.

6.5.1 MAPGTx - USAGE. Use one of the four statements
CALL MAPGTC (WHCH, CVAL)
CALL MAPGTI (WHCH, IVAL)
CALL MPGTL (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).

6.5.2 MAPGTx - 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 JLTS from the last call to MAPSET. The default value is 'MA'.</td>
</tr>
<tr>
<td>Error</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>PROjection</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>
</tbody>
</table>
Pn I, R "n" 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.

Rotation I, R The value of ROTA from the last call to MAPROJ. The default value is zero.

XLeft R The parameters XLLOW, XROW, YBOW, and YTOW from the last call to MAPPOS. Defaults are .05, .95, .05, and .95, respectively.

XRight YBottom YTop

6.6 MAPINT. Initialization.

6.6.1 MAPINT - 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 MAPGL, indicates whether or not initialization is required at a given time. As of now (March, 1985), no parameter change by means of a call to MAPSTx forces re-initialization; only calls to MAPPOS, MAPROJ, and MAPSET do.

6.6.2 MAPINT - ARGUMENTS. None.

6.7 MAPIQ. Terminate a string of calls to MAPIT.

6.7.1 MAPIQ - 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 intensity, dash pattern, color, etc.

6.7.2 MAPIQ - ARGUMENTS. None.

6.8 MAPIT (RLAT, RCON, IFST). Draws lines on a map.
6.8.1 MAPIT - USAGE. 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 (which see, above) to flush its buffers before a STOP, a CALL FRAME, or a call to change the intensity, the color, etc.

Points in two contiguous pen-down calls to MAPIT should not be far apart on the globe.

6.8.2 MAPIT - 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.

6.9 MAPLBL. Labels the map.

6.9.1 MAPLBL - 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.
6.9.2 MAPLBL - ARGUMENTS. None.

6.10 MAPLOT. Draws geographical outlines.

6.10.1 MAPLOT - 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.

6.10.2 MAPLOT - ARGUMENTS. None.

6.11 MAPPOS (XLOW,XROW,YBOW,YTOW). Positions the map on the plotter frame.

6.11.1 MAPPOS - 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).

6.11.2 MAPPOS - ARGUMENTS. REAL XLOW,XROW,YBOW,YTOW.

XLOW specifies the position of the left edge of the window. Default value 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.
6.12 MAPROJ (JPRJ, PLAT, PLON, ROTA). Set the projection to be used.

6.12.1 MAPROJ - USAGE. The statement

    CALL MAPROJ (JPRJ, PLAT, PLON, ROTA)

sets EZMAP parameters specifying the projection to be used for subsequent maps.

6.12.2 MAPROJ - 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 MAPSTR.
PIAT, 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': PIAT 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 counter-clockwise, 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".

6.13 MAPRS. Recalls SET.

6.13.1 MAPRS - 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.

6.14 MAPRST (IFNO). Restores the state of EZMAP saved by an earlier call to MAPSAV.

6.14.1 MAPRST - 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.

6.14.2 MAPRST - 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.

6.15 MAPSAV (IFNO). Saves the current state of EZMAP for later restoration by MAPRST.

6.15.1 MAPSAV - 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.

6.15.2 MAPSAV - 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.

6.16 MAPSET (JLTS,PLM1,PLM2,PLM3,PLM4). To specify the rectangular portion of the u/v plane to be drawn.

6.16.1 MAPSET - USAGE. The statement

    CALL MAPSET (JLTS,PLM1,PLM2,PLM3,PLM4)

specifies what portion of the u/v plane is to be plotted.

6.16.2 MAPSET - 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.

- \textbf{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.

- \textbf{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''.

- \textbf{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.

---

6.17 \textbf{MAPSTx (WHICH,xVAL).} To set the values of certain EZMAP parameters.

6.17.1 \textbf{MAPSTx - USAGE.} Use one of the four statements

\begin{verbatim}
CALL MAPSTC (WHICH,CVAL)
CALL MAPSTI (WHICH,IVAL)
CALL MAPSTL (WHICH,LVAL)
CALL MAPSTR (WHICH,RVAL)
\end{verbatim}

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 \texttt{`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.

6.17.2 MAPSTX - ARGUMENTS. CHARACTER*2 CVAL; INTEGER IVAL; LOGICAL LVAL; REAL RVAL.

WHICH 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 WHICH - 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>WHICH</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<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>D</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 &quot;RE&quot;, below).</td>
</tr>
<tr>
<td>D</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. “D” 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>
</tbody>
</table>
ELLiptical I, L
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).

GD R
The distance between points used to draw the grid, in degrees. The default value is 1.; user values must fall between .001 and 10.

GRid I, R
The desired grid spacing. A zero suppresses the grid. The default is 10 degrees.

In I
"n" is an integer between 1 and 7. Each "In" specifies the intensity of some portion of the map. Values are in the range 0-255. The defaults are:

<table>
<thead>
<tr>
<th>n</th>
<th>Use</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>perimeter</td>
<td>240</td>
</tr>
<tr>
<td>2</td>
<td>grid</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>labels</td>
<td>210</td>
</tr>
<tr>
<td>4</td>
<td>limbs</td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>continents</td>
<td>240</td>
</tr>
<tr>
<td>6</td>
<td>U.S. states</td>
<td>180</td>
</tr>
<tr>
<td>7</td>
<td>countries</td>
<td>210</td>
</tr>
</tbody>
</table>

Label I, L
If true (non-zero), label the meridians and the poles. The default is false (zero).

LS I
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.

MW I, R
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).

Outline C
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".
PERim I,L If true (non-zero), draw the perimeter. The default is true (non-zero).

RESolution I,R The width of the target plotter, in plotter units. The default value is 4096.

SAtellite I,R 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', below.

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 counter-clockwise.

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.

6.18 MAPTRN (RLAT,RLON,UVAL,VVAL). To project points.

6.18.1 MAPTRN - 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).

6.18.2 MAPTRN - 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, WAL) is the projection in the u/v plane of (RLAT, RLON). The units of UVAL and WAL depend on the projection.

If the point is not projectable, UVAL is returned equal to 1.EL2. 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.

6.19 MAPUSR (IPRT). To change the appearance of the various parts of a map.

6.19.1 MAPUSR - USAGE. EZMAP executes the statement

```fortran
CALL MAPUSR (IPRT)
```

just before and just after each portion of a map is drawn. The default GKS version of MAPUSR does nothing. For the sake of efficiency, the non-GKS default version looks like this:

```fortran
SUBROUTINE MAPUSR (IPRT)
  SAVE IDSH,JDISH
  IF (IPRT.EQ.2) THEN
    CALL GETOPT ('DP',IDSH)
    CALL MAPGPI ('DA',JDISH)
    CALL OPTN ('DP',JDISH)
    CALL DASHD (IOR(ISHIFT(32767,1),1),0,0,0)
  ELSE IF (IPRT.EQ.-2) THEN
    CALL OPTN ('DP',IDSH)
    CALL DASHD (JDISH,0,0,0)
  END IF
  RETURN
END
```

which sets the DASHD pattern for grid lines to "solid" and does an OPTN call to make the translator generate the desired pattern. This is faster and creates shorter metacode files.

A user-supplied version of MAPUSR may set/reset the dotting parameter "DL", the DASHCHAR dash pattern, the intensity, the color, etc., so as to achieve a desired effect.

6.19.2 MAPUSR - 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.

6.20 MAPVEC (RLAT,RLON). Draws lines on the map produced by a call to MAPDRW - used in conjunction with MAPFST.

6.20.1 MAPVEC - USAGE. The statement

    CALL MAPVEC (RLAT,RLON)

is equivalent to the statement

    CALL MAPIT (RLAT,RLON,1)

See the description of MAPIT.

6.20.2 MAPVEC - ARGUMENTS. REAL RLAT,RLON.

RLAT and RLON are defined as for MAPIT (which see).

6.21 SUPCON (RLAT,RLON,UVAL,VVAL). 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.

6.21.1 SUPCON - USAGE. The statement

    CALL SUPCON (RLAT,RLON,UVAL,VVAL)

is exactly equivalent to the statement

    CALL MAPTRN (RLAT,RLON,UVAL,VVAL)

6.21.2 SUPCON - ARGUMENTS. REAL RLAT,RLON,UVAL,VVAL.

RLAT, RLON, UVAL, and VVAL are defined as for MAPTRN (which see).
6.22 SUPMAP (...) An implementation of the routine from which EZMAP grew.

6.22.1 SUPMAP - 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 (which see) 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.

6.22.2 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 speci-
fies 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 (which see).
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 (which see), 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.
7. EXAMPLES

This write-up contains the FORTRAN text and Dicommed 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 labelling 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.

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.
7.1 **EXAMPLE 1.** The United States is mapped using a Lambert conformal conic.

```
THE U.S. ON A LAMBERT CONFORMAL CONIC
```

```
PROGRAM EXMPL1
C
C The program EXMPL1 produces a map of the U.S., using a Lambert conformal conic.
C
C Define the label for the top of the map.
C
    CHARACTER*37 PLBL
C
EXAMPLES
```
DATA PBL / 'THE U.S. ON A LAMBERT CONFORMAL CONIC' /

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.)
C CALL PWRT (.5, .925, 'PLBL37, 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 Done.
C STOP

END

SUBROUTINE BNDARY

C Routine to draw the plotter-frame edge.
C CALL PLOTIT ( 0, 0, 0)
C CALL PLOTIT (32767, 0, 1)
C CALL PLOTIT (32767, 32767, 1)
C CALL PLOTIT ( 0, 32767, 1)
C CALL PLOTIT ( 0, 0, 1)
RETURN
END
7.2 EXAMPLE 2. A stereographic view of the international boundaries in Africa, jazzed up by using an elliptical frame.

PROGRAM EXMPL2

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
EXAMPLES - 3-4 -
DATA PLBL / "CAN YOU NAME THE COUNTRIES" /
C
C Use an elliptical perimeter.
C
CALL MAPSTI ('EL',1)
C
Dot the outlines, using dots half as far apart as the default.
C
CALL MAPSTI ('DO',1)
CALL MAPSTI ('DD',6)
C
Show continents and international boundaries.
C
CALL MAPSTC ('OU', 'PO')
C
Use a stereographic projection.
C
CALL MAPROJ ('ST',0.,0.,0.)
C
Specify where two corners of the map are.
C
CALL MAPSET ('CO', -38., -28., 40., 62.)
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, .975, PLBL, 26, 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
Done.
C
STOP
C
END
SUBROUTINE BNDARY
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 PLOTT ( 0, 0, 1)
RETURN
END
7.3 **EXAMPLE 3.** A Mercator projection of the world, using a version of MAPEOS which scrubs all but the principal land masses (with a tip of the hat to Cicely Ridley). The segment-number information used in MAPEOS was obtained by running the program shown in example 9.

---

**PROGRAM EXMPL3**

C
C Produce a Mercator projection of the whole globe, using
C simplified continental outlines. See the routine MAPEOS,
C below.
C
C Define the label for the top of the map.

C
CHARACTER*46 PLBL
DATA PLBL/'SIMPLIFIED CONTINENTS ON A MERCATOR PROJECTION'/
CALL SUPMAP (9,0.,0.,0.,0.,0.,0.,0.,1,15,2,0,IERR)

C Put the label at the top of the plot.

CALL SET (0.,1.,0.,1.,1.,0.,1.,1)
CALL PWRIT (.5,.975,PLBL,46,2,0,0)

C Draw a boundary around the edge of the plotter frame.

CALL BNDARY

C Advance the frame.

CALL FRAME

C
STOP

END

SUBROUTINE MAPEGO (NOUT,NSEG,IGID,NPTS,PNTS)

DIMENSION PNTS(*)

C This version of MAPEGO uses segment numbers for the outline
dataset 'G0' to suppress all but the major global land masses.
In honor of Cicely Ridley, the British Isles (segments 203 and
204) are included.

IF ((NOUT.EQ.1).AND.
+ (NSEG.LT. 1.OR.NSEG.GT. 6).AND.
+ (NSEG.LT. 16.OR.NSEG.GT. 30).AND.
+ (NSEG.LT.112.OR.NSEG.GT.113).AND.
+ (NSEG.LT.117.OR.NSEG.GT.139).AND.
+ (NSEG.LT.203.OR.NSEG.GT.204).AND.
+ (NSEG.LT.221.OR.NSEG.GT.223) ) NPTS=0

C Done.

RETURN

END

SUBROUTINE BNDARY

C Routine to draw the plotter-frame edge.

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
7.4 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 projection for the equatorial belt.

PROGRAM EXMPL4

This program produces a single frame, with polar stereographic views of the poles and a Mercator projection of the rest.

Define the label for the top of the map.

CHARACTER*26 PLBL

EXAMPLES - 40 - EZMAP
DATA PLBL / 'ONIA TERRA IN PARTES TRES' /  
C  Set the outline-dataset parameter.  
C  CALL MAPSTC ('OU','PO')  
C  Use dotted outlines and move the dots a little closer together  
C  than normal.  
C  CALL MAPSTI ('DO',1)  
C  CALL MAPSTI ('DD',8)  
C  Crank up the intensity of the international boundaries.  
C  CALL MAPSTI ('I7',225)  
C  Do the Mercator projection of the equatorial belt first.  
C  CALL MAPPOS (.05,.95,.05,.5)  
C  CALL MAPROJ ('ME',0.,0.,0.)  
C  CALL MAPSET ('LT',-3.1416,3.1416,-1.5708,1.5708)  
C  CALL MAPDRW  
C  Switch to an elliptical (in this case, circular) boundary.  
C  CALL MAPSTI ('EL',1)  
C  Do a polar stereographic view of the North Pole ...  
C  CALL MAPPOS (.07,.48,.52,.93)  
C  CALL MAPROJ ('ST',90.,0.,-90.)  
C  CALL MAPSET ('AN',30.,30.,30.,30.)  
C  CALL MAPDRW  
C  and then a similar view of the South Pole.  
C  CALL MAPPOS (.52,.93,.52,.93)  
C  CALL MAPROJ ('ST',-90.,0.,90.)  
C  CALL MAPSET ('AN',30.,30.,30.,30.)  
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)  
C  CALL PWRT (.5,.975,PLBL,26,2,0,0)  
C  Draw a boundary around the edge of the plotter frame.  
C  CALL BNDARY  
C  Advance the frame.  
C
CALL FRAME
C
C Done.
C
STOP
C
END
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
7.5 **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 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 Set the outline-dataset parameter.
C
CALL MAPSTC ("OU", "CO")
C
Put meridians and parallels every 15 degrees.
C
CALL MAPSTI ("GR", 15)
C
Reduce the label size.
C
CALL MAPSTI ("LS", 0)
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
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 Mercator.

CALL MAPPOS (.75625,.975,.3875,.60625)
CALL MAPROJ ('ME',0.,0.,0.)
CALL MAPDRW

C Cylindrical equidistant.

CALL MAPPOS (.025,.4875,.13125,.3625)
CALL MAPROJ ('CE',0.,0.,0.)
CALL MAPDRW

C Mollweide type.

CALL MAPPOS (.5125,.975,.13125,.3625)
CALL MAPROJ ('MO',0.,0.,0.)
CALL MAPDRW

C Put the label at the top of the plot ...

CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PWRT (.5,.925,PLBL,37,2,0,0)

C and the labels under each sub-plot.

CALL PWRT (.134375,.61875,'LAMBERT CONFORMAL CONIC',
+        23,0,0,0)
CALL PWRT (.378125,.61875,'STEREGRAPHIC',13,0,0,0)
CALL PWRT (.621875,.61875,'ORTHOGRAPHIC',12,0,0,0)
CALL PWRT (.865625,.61875,'LAMBERT EQUAL-AREA',18,0,0,0)
CALL PWRT (.134375,.375,'GNOMONIC',8,0,0,0)
CALL PWRT (.378125,.375,'AZIMUTHAL EQUIDISTANT',21,0,0,0)
CALL PWRT (.621875,.375,'SATELLITE-VIEW',14,0,0,0)
CALL PWRT (.865625,.375,'MERCATOR',8,0,0,0)
CALL PWRT (.25625,.11875,'CYLINDRICAL EQUIDISTANT',
+        23,0,0,0)
CALL PWRT (.74375,.11875,'MOLLWEIDE TYPE',14,0,0,0)

C Draw a boundary around the edge of the plotter frame.

CALL BNDARY

C Advance the frame.

CALL FRAME

C Done.

STOP
SUBROUTINE MPECS (NOUT, NSEG, IGID, NPTS, PNTO)

DIMENSION PNTO(*)

This version of MAPEOS uses segment numbers for the outline dataset "CO" to suppress all but the major global land masses.

IF ((NOUT.EQ.1) .AND.
    + (NSEG.LT. 1 .OR. NSEG.GT. 6) .AND.
    + (NSEG.LT. 16 .OR. NSEG.GT. 30) .AND.
    + (NSEG.LT. 112 .OR. NSEG.GT. 113) .AND.
    + (NSEG.LT. 117 .OR. NSEG.GT. 139) .AND.
    + (NSEG.LT. 221 .OR. NSEG.GT. 223) ) NPTS=0

RETURN

END

SUBROUTINE BNDARY

Routine to draw the plotter-frame edge.

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

EXAMPLES - 46 - EZMAP
7.6 **EXAMPLE 6.** Shows the effect of changing the rotation angle ROTA. Note that the satellite is positioned over Kansas.

![Diagram with four views of the Earth at different rotation angles](image)

**THE EARTH IS SPINNING. TOTO**

**ROTA = 0**

**ROTA = 90**

**ROTA = 180**

**ROTA = 270**

**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
DATA PLBL / 'THE EARTH IS SPINNING, TOTO' /

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 ('SI', 15)
C Turn off the perimeter and reduce the number of grid lines.
C CALL MAPSTI ('PB', 0)
C 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)
C CALL MAPPROJ ('SV', 38., -98., 0.)
C CALL MAPSTI ('S2', 135)
C CALL MAPDRW
C Repeat, but rotate by 90 degrees and look to the upper right.
C CALL MAPPOS (.525, .95, .525, .95)
C CALL MAPPROJ ('SV', 38., -98., 90.)
C CALL MAPSTI ('S2', 45)
C CALL MAPDRW
C Repeat, but rotate by 180 degrees and look to the lower left.
C CALL MAPPOS (.05, .475, .05, .475)
C CALL MAPPROJ ('SV', 38., -98., 180.)
C CALL MAPSTI ('S2', -135)
C CALL MAPDRW
C Repeat, but rotate by 270 degrees and look to the lower right.
C CALL MAPPOS (.525, .95, .05, .475)
C CALL MAPPROJ ('SV', 38., -98., 270.)
C CALL MAPSTI ('S2', -45)
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.)
C CALL PWRT (.5, .975, PLBL, 27, 2, 0, 0)
C and the ones below each sub-plot.

CALL PWRIT (.2625,.5, "ROTA = 0°,8,1,0,0)
CALL PWRIT (.7375,.5, "ROTA = 90°,9,1,0,0)
CALL PWRIT (.2625,.025, "ROTA = 180°,10,1,0,0)
CALL PWRIT (.7375,.025, "ROTA = 270°,10,1,0,0)

C Draw a boundary around the edge of the plotter frame.
CALL BNDARY

C Advance the frame.
CALL FRAME

C Done.
STOP

END

SUBROUTINE BNDARY

C Routine to draw the plotter-frame edge.

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
7.7 Example 7. It is difficult to write a general algorithm for labelling 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 EXAMPLE

C This program produces a stereographic view of the North Pole, with labelled meridians.

EXAM PLES - 50 -
C Define the label for the top of the map.
C
CHARACTER*32 PLBL
C
DATA PLBL / 'MERIDIONAL LABELS ON A POLAR MAP' /
C
Move the map a little to provide more room for labels.
C
CALL MAPPOS (.075,.925,.05,.90)
C
Use an elliptical (circular, in this case) perimeter.
C
CALL MAPSTI ('EL','1)
C
Show continents and international boundaries.
C
CALL MAPSTC ('OU','PO')
C
Use a stereographic projection, centered at the North Pole.
C
CALL MAPROJ ('ST',90.,0.,-100.)
C
Specify the angular distances to the edges of the map.
C
CALL MAPSET ('AN',80.,80.,80.,80.)
C
Draw the map.
C
CALL MAPDRW
C
Call a routine to label the meridians. This routine is not
C
a part of EZMAP; the code is given below.
C
CALL MAPLBM
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,32,2,0,0)
C
Draw a boundary around the edge of the plotter frame.
C
CALL BNDARY
C
Advance the frame.
C
CALL FRAME
C
Done.
C
STOP
C
END
EZMAP - 51 - EXAMPLES
SUBROUTINE MAPLBM

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.

Necessary local declarations.

CHARACTER*2 PROJ
CHARACTER*3 CHRS
CHARACTER*4 CHLB

See if the conditions required for MAPLBM to work are met.

The projection must be azimuthal, ...

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

the pole latitude must be +90 degrees or -90 degrees, ...

CALL MAPGTR ("PT", PLAT)
IF (ABS(PLAT).LT.89.9999) RETURN

the perimeter must be elliptical, ...

CALL MAPGTI ("EL", IELP)
IF (IELP.EQ.0) RETURN

the values used in the SET call must define a circle, ...

CALL GETSET (ILEW, IREW, IBEW, ITEW, ULEW, UREW, VB EW, VTEW, LNLG)
IF (ULEW-UREW.GT.0.0001.OR.VBEW-VTEW.GT.0.0001) RETURN
IF (ILEW-IT EW.GT.0.0001.OR.VBEW+UREW.GT.0.0001) RETURN

and the grid spacing must be an integral divisor of 180.

CALL MAPGTR ("GR", GRID)
IF (AMOD(GRID,1.).NE.0.OR.
+ MOD(180,IFIX GRID).NE.0) RETURN

All conditions are satisfied. Label the meridians.

Collect the necessary information.

IGG=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(IREW-IBEW)) *(UREW-VBEW)
HOLB=HOCH/1.5

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
C
C Compute the width of the label.
C
WDLB=FLOAT(NCHS)*WOCH
C
C Find the angle at which the labelled meridian lies on the plot.
C
IF (PLAT.GT.0.) THEN
  ANOD=FLOAT(I-90)-PLON-ROTA
ELSE
  ANOD=FLOAT(90-I)+PLON-ROTA
END IF
C
C Reduce the angle to the range from -180 to +180 and
C find its equivalent in radians.
C ANGD=ANGD-S SIGN(180.,ANGD+180.)
   + SIGN(180.,180.-ANGD)
ANGR=.017453292519943*ANGD
C
C Figure out where the end of the meridian is.
C
XEND=UREW*COS(ANGR)
YEND=VREW*SIN(ANGR)
C
C Extend the meridian a little to make a tick mark.
C
CALL LINE (XEND,YEND, 1.015*XEND, 1.015*YEND)
C
C Compute a center position for the label which puts its nearest
C edge at a fixed distance from the perimeter. First, compute
C the components (DELX,DELY) of the vector from the center of the
C 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
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
C Then, solve (for FMUL) the following equation:
C
C SQRT((FMUL*XEND+DELX)**2+(FMUL*YEND+DELY)**2))=1.02*UREW
C
C which expresses the condition that the corner of the box

EXAMPLES - 54 - EZMAP
C nearest the circular perimeter should be at a distance of
C \(1.02 \times (\text{the radius of the perimeter})\) away from the center of
C the plot.
C
A = XEND*YEND + YEND*XEND
B = 2. \times (XEND*DELY+YEND*DELY)
C = DELX*DELY*DELY - 1.0404*UREW*UREW

FMUL = \((-B+\sqrt{B^2-4. \times A \times C})/(2. \times A)\)

C Draw the label.
C
CALL PWRT (FMUL*XEND, FMUL*YEND, CHLB, NC, ICSZ, 0, 0)

C End of loop.
C
101 CONTINUE
C
C Done.
C
RETURN
C
C Format
C
1001 FORMAT (I3)
C
END

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
This example demonstrates how a user-provided routine MAPUSR may be employed to change the way in which portions of a map are drawn.

ILLUSTRATING THE USE OF MAPUSR

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

DATA PLBL / 'ILLUSTRATING THE USE OF MAPUSR' /

Weird up the projection a little.

CALL SUPMAP (9.,0.,0.,90.,0.,0.,0.,0.,15.,2,0.,IERR)

Put the label at the top of the plot.

CALL SET (0.,1.,0.,0.,1.,0.,0.,1.,1)
CALL PWRIT (.5,.975,PLBL,30,2,0,0)

Draw a boundary around the edge of the plotter frame.

CALL BNDARY

Advance the frame.

CALL FRAME

STOP

END

SUBROUTINE MAPUSR (IPRT)

This version of MAPUSR forces the grid lines to be dotted and
the outlines to be dashed.

Certain local parameters must be saved from call to call.

SAVE INTN,IDTF,IDBD

If IPRT is positive, a part is about to be drawn. Save the
current intensity, the dotted/solid flag, and/or the distance
between dots and then reset them and/or the dash pattern.

IF (IPRT.GT.0) THEN
  IF (IPRT.EQ.2) THEN
    CALL GETOPT ('IN',INTN)
    CALL MAPGTI ('DL',IDTF)
    CALL MAPGTI ('DD',IDBD)
    CALL OP'TN ('IN',255)
    CALL MAPSTI ('DL',1)
    CALL MAPSTI ('DD',12)
  ELSE IF (IPRT.EQ.5) THEN
    CALL GETOPT ('IN',INTN)
    CALL MAPGTI ('DL',IDTF)
    CALL OP'TN ('IN',150)
    CALL MAPSTI ('DL',0)
    CALL DASHD (52525B,0,0,0)
  END IF

END
ELSE

C Otherwise, a part has just been drawn. Restore saved settings
C and/or select a solid dash pattern.
C
C IF (IPRT.EQ.-2) THEN
  CALL OPTN ('IN', INTN)
  CALL MAPSTI ('DL', IDTF)
  CALL MAPSTI ('DD', IDBD)
ELSE IF (IPRT.EQ.-5) THEN
  CALL OPTN ('IN', INTN)
  CALL MAPSTI ('DL', IDTF)
  CALL DASHD (17777B, 0, 0, 0)
END IF
C
C END IF
C
C Done.
C
C RETURN
C
C END
C
SUBROUTINE BNDARY
C
C Routine to draw the plotter-frame edge.
C
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
7.9 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. The program can produce 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.) Note that Antarctica is composed of segments 1 through 6, North and South America of segments 16 through 30, Eurasia/Africa of segments 117 through 139, Australia of segments 221 through 223, and Greenland of segments 112 through 113.

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 The program EXMPL9 produces a set of plots showing the numbers of all the segments in a chosen EZMAP outline dataset. Certain variables must be set in data statements at the beginning of the program. In each of the seven places marked off by rows of dashes, un-comment the first card to do outline dataset 'CO', the second to do 'US', the third to do 'PO', and the fourth to do 'PS'.

C The common block LIMITS communicates values between TESTIT and the routines MAPEDS and MOVEIT.
COMMON /LIMITS/ ICSZ, ULEW, UREW, VBEW, VIEW, DELU, DELV, NSAV

C Select the outline dataset.
C
CHARACTER*2 OUTD
C
DATA OUTD / "CO" /
DATA OUTD / "US" /
DATA OUTD / "PO" /
DATA OUTD / "PS" /
C
C Select the projection type.
C
CHARACTER*2 PROJ
C
DATA PROJ / "ME" /
DATA PROJ / "LC" /
DATA PROJ / "ME" /
DATA PROJ / "ME" /
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
DATA LMTS / "MR" /
DATA LMTS / "CO" /
DATA LMTS / "MR" /
DATA LMTS / "MR" /
C
C Select the values to be put in the limits arrays.
C
DIMENSION PL1(2), PL2(2), PL3(2), PL4(2)
C
DATA PL1(1), PL2(1), PL3(1), PL4(1) / 0., 0., 0., 0. /
DATA PL1(1), PL2(1), PL3(1), PL4(1) / 22.6, -120.4, 46.9, -64.2 /
DATA PL1(1), PL2(1), PL3(1), PL4(1) / 0., 0., 0., 0. /
Select values determining how the whole map is to be carved up into little maps. ILIM is the number of divisions of the horizontal axis, JLIM the number of divisions of the vertical axis. (If all the labels are put on a single map, the result is confusing.)

DATA ILIM, JLIM / 2, 2 /
DATA ILIM, JLIM / 3, 2 /
DATA ILIM, JLIM / 2, 2 /
DATA ILIM, JLIM / 6, 6 /

Define a variable to hold the plot label.

CHARACTER*38 PLBL

DATA PLBL / 'SEGMENT NUMBERS FOR OUTLINE DATASET XX' /

Finish the plot label.

PLBL(37:38) = OUTD

Set the character size; 6 is about the smallest usable value.

ICSZ=6

Set the outline-dataset parameter.

CALL MAPSTC ("OU", OUTD)

Set the projection-type parameters.

CALL MAPROJ (PROJ, PLAT, PLON, ROTA)

Set the limits parameters.

CALL MAPSET (LMTS, PL1, PL2, PL3, PL4)

Force the intensities of all the outlines to the same value.

CALL MAPSTI ("I5", 160)
CALL MAPSTI ("I6", 160)
CALL MAPSTI ("I7", 160)

CALL MAPINT

CALL GETSET (ILEM, IREM, IBEM, ITEM, ULEM, UREM, VBEM, VTEM, NLG)

Now, plot a set of maps which are subsets of the whole map.

DO 102 I = 1, ILIM
    DO 101 J = 1, JLIM
        CALL MAPSET ("LI",
                    + ULEM*(UREM-ULEM)*FLOAT(I-1)/FLOAT(ILIM),
                    + UREM*(UREM-ULEM)*FLOAT(I )/FLOAT(ILIM),
                    + VBEM*(VTEM-VBEM)*FLOAT(J-1)/FLOAT(JLIM),
                    + VBEM*(VTEM-VBEM)*FLOAT(J )/FLOAT(JLIM))
    
Re-initialize EZMAP with the new limits.

CALL MAPINT

CALL GETSET (ILEW, IREW, IBEW, ITEW,
             + ULEW, UREW, VB EW, VTEW, NLG)

Compute quantities required by MAPEOS and MOVEIT to position labels.

DELU=3.5*(FLOAT(ICSZ)/FLOAT(IREW-ILEW))*(UREW-ULEW)
DELV=2.0*(FLOAT(ICSZ)/FLOAT(ITEW-IBEW))*(VTEW-VB EW)
NSAV=0

Draw the outlines.

CALL MAPLOT

Put a label at the top of the plot.

CALL SET (0., 1.0, 0., 1., 0., 1., 0., 1., 1)
CALL PWRT (.5, .975, PLBL, 38, 2, 0, 0)

Draw a boundary around the edge of the plotter frame.

CALL BNDARY

Advance the frame.

CALL FRAME

EXAMPLES - 66 -
SUBROUTINE MAPEOS (NOUT, NSEG, IGID, INPTS, PNTS)

C
DIMENSION PNTS(*)
C
C This version of MAPEOS marks each segment of the map with its
C segment number. The resulting map may be used to set up other
C versions of MAPEOS which plot selected segments.
C
C The common block LIMITS communicates values between TESTIT and
C the routines MAPEOS and MOVEIT.
C
COMMON /LIMITS/ ICSZ, ULEW, UREW, VBEW, VTEW, DELU, DELV, NSAV
C
C Define local variables to hold the character-string form
C of the segment number.
C
CHARACTER*4 CSG1, CSG2
C
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
C If the center is visible, label it.
C
IF (UCEN .GE. ULEW .AND. UREN .LE. UREW .AND.
+ VCEN .GE. VBEW .AND. VCEN .LE. VTEW) THEN
C
C Generate a character string representing the value of the
C segment number.
C
WRITE (CSG2, 1001) NSEG
NFCH = 4 - IFIX(ALOG10(REAL(NSSEG) + 1.5))
NOHS = 5 - NFCH
CSG1 = CSG2(NOHS:NFCH)
C
C Find out where the two points on either side of the center are.
C
MPTS = MAX0(NPTS/2, 1)
CALL MAPTRN (PNTS(2*MPTS-1), PNTS(2*MPTS), UCML, VCML)
C
MPTS = MIN0(NPTS/2+1, 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.

ULAB = UCEN - SIGN (.1428 * DELU * FLOAT (NCHS), UCMI + UCP1 - 2 * UCEN)
VLAB = VCEN - SIGN (.3333 * DELV, VCM1 + VCP1 - 2 * VCEN)

C Move the label as necessary to avoid its being on top of any
C previous label.

CALL MOVEIT (ULAB, VLAB)

C Write out the character string and connect it to the
C segment with a straight line.

CALL LINE (UCEN, VCEN,
          + ULAB - SIGN (.1428 * DELU * FLOAT (NCHS), ULAB - UCEN),
          + VLAB - SIGN (.3333 * DELV, VLAB - VCEN))

CALL PWRTIT (ULAB, VLAB, CSG1, NCHS, ICSZ, 0, 0)

END IF

C Done.

RETURN

C Format.

1001 FORMAT (I4)

END

SUBROUTINE MOVEIT (ULAB, VLAB)

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 The common block LIMITS communicates values between TESTIT and
C the routines MAPEOS and MOVEIT.

C ULEW, UREW, VB EW, and VTEW specify the u/v limits of the window
C in which the map was drawn. DELU and DELV are the minimum
C allowable distances between segment-label centers in the u
C and v directions, respectively. NSAV is the number of label
C 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)

EXAMPLES - 68 - EZ:MP
USA and VSAV must maintain their values between calls.

SAVE USAV, VSAV

Zero the variables which control the generation of a spiral.

IGN1=0
IGN2=2

Check for overlap at the current position.

101 DO 102 I=1,NSAV
   IF (ABS(ULAB-USA(I)).LT.DELU.AND.
     + ABS(VLAB-VSA(I)).LT.DELV) GO TO 103
102 CONTINUE

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

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 SUBROUTINE BNDARY

Routine to draw the plotter-frame edge.

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
7.10 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 C Define a data array.
C DIMENSION X/CD(224)
C C Define the centers and the expansion/shrinkage factors for C 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 Fill the data array.
C
READ 1001, XCYD
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 high-intensity dotted lines.
C
CALL OPTN ('IN', 255)
CALL MAPSTI ('DL', 1)
C
C Draw some curves.
C
DO 102 I=1, 4
C
IFST=0
C
DO 101 J=1, 112
IF (XCYD(2*J-1).EQ.0.) THEN
IFST=0
ELSE
FLON=FLOC(I)+FMUL(I)*(XCYD(2*J-1)-15.)
FLAT=FLAC(I)+FMUL(I)*(XCYD(2*J)-15.)
CALL MAPIT (FLAT, FLON, IFST)
IFST=1
END IF
101 CONTINUE
C 102 CONTINUE
C
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.
CALL FRAME

STOP

C Format.

1001 FORMAT (14E5.0)

END

SUBROUTINE BNDARY

C Routine to draw the plotter-frame edge.

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
GRIDAL
SUBROUTINE GRIDAL(MAJRX, MINRX, MAJRY, MINRY, IXLAB, IYLAB, IGPH, X, Y)

LATEST REVISION July, 1985

PURPOSE This is a package of routines for drawing graph paper, axes, and other backgrounds.

USAGE Each user entry point in this package (GRID, GRIDL, PERIM, PERIML, HALFAX, LABMOD, TICK4, and GRIDAL) will be described separately below. First, however, we will discuss how major and minor divisions in the graph paper are handled by all entries which use them.

GRIDAL, GRID, GRIDL, PERIM, PERIML, and HALFAX have arguments MAJRX, MINRX, MAJRY, MINRY which control the number of major and minor divisions in the graph paper or perimeters. The number of divisions refers to the holes between lines rather than the lines themselves. This means that there is always one more major division line than the number of major divisions. Similarly, there is one less minor division line than minor divisions (per major division.)

MAJRX, MAJRY, MINRX, MINRY have different meanings depending upon whether log scaling is in effect (set via SETUSV or SET in the SPPS package.)

For linear scaling, MAJRX and MAJRY specify the number of major divisions along the X-axis or Y-axis respectively, and MINRX and MINRY specify the number of minor divisions per major division.

For log scaling along the X-axis each major division occurs at a factor of $10^{**MAJRX}$ times the previous division. For example, if the minimum X-axis value is 3., and the maximum X-axis value is 3000., and MAJRX is 1, then major divisions will occur at 3., 30., 300., and 3000. Similarly for MAJRY. If log scaling is in effect on the X-axis and MINRX.LE.10, then there are nine minor divisions between each major division. For example, between 3. and 30. there would be a minor division at 6., 9.,
12,...,27. If log scaling is in effect on the X-axis and MINRX.GT.10, then there will be no minor subdivisions. MINRY is treated in the same manner as MINRX.

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGRIMJ</td>
<td>Color index for normal (major) intensity lines.</td>
</tr>
<tr>
<td>IGRIMN</td>
<td>Color index for low intensity lines.</td>
</tr>
<tr>
<td>IGRITX</td>
<td>Color index for text (labels.)</td>
</tr>
</tbody>
</table>

We now describe each entry in this package.

---

**SUBROUTINE GRID**

**PURPOSE**
To draw graph paper.

**USAGE**
CALL GRID (MAJRX,MINRX,MAJRY,MINRY)

**DESCRIPTION**
This subroutine draws graph lines in the portion of the plotter specified by the current viewport setting with the number of major and minor divisions as specified by the arguments.

---

**SUBROUTINE GRIDAL**

**PURPOSE**
A general entry point for all background routines with the option of line labelling on each axis.

**USAGE**
CALL GRIDAL (MAJRX,MINRX,MAJRY,MINRY,IXLAB,IYLAB,IGPH,X,Y)

**ARGUMENTS**
- MAJRX,MINRX,MAJRY,MINRY
  Major and minor axis divisions as described in the USAGE section of the package documentation above.

- IXLAB,IYLAB
  Same as the parameters to LABMOD.

- IGPH
  Flag for background type:

April, 1986
<table>
<thead>
<tr>
<th>IGPH</th>
<th>X-axis background</th>
<th>Y-axis background</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>

X,Y
World coordinates of the intersection of the axes
if IGPH=10.

---

**SUBROUTINE GRIDL**

**PURPOSE**
To draw graph paper.

**USAGE**
CALL GRIDL (MAJRX,MINRX,MAJRY,MINRY)

**DESCRIPTION**
This subroutine behaves exactly as GRID, but each major division is labeled with its numerical value.

---

**SUBROUTINE HALFAX**

**PURPOSE**
To draw orthogonal axes.

**USAGE**
CALL HALFAX (MAJRX,MINRX,MAJRY,MINRY,X,Y,IXLAB,IYLAB)

**DESCRIPTION**
This subroutine draws orthogonal axes intersecting at coordinate (X,Y) with optional labeling options as specified by IXLAB and IYLAB.

**ARGUMENTS**
MAJRX,MINRX,MAJRY,MINRY
Major and minor division specifications as per the description in the package USAGE section above.

X,Y
World coordinates specifying the intersection point of the X and Y axes.

IXLAB,IYLAB (Integers)
Flags for axis labels:

IXLAB = -1 No X-axis drawn

---

April, 1986 3 GRIDAL
No X-axis labels
= 0 X-axis drawn
No X-axis labels
= 1 X-axis drawn
X-axis labels

IYLAB = -1 No Y-axis drawn
No Y-axis labels
= 0 Y-axis drawn
No Y-axis labels
= 1 Y-axis drawn
Y-axis labels

SUBROUTINE LABMOD

PURPOSE
To allow more complete control over the appearance of the labels on the background plots.

USAGE
CALL LABMOD (FMTX,FMTY,NUMX,NUMY,ISIZX,ISIZY,IXDEC,IYDEC,IXOR)

DESCRIPTION
This subroutine presets parameters for the other background routines in this package. LABMOD itself does no plotting and it must be called before the background routines for which it is presetting parameters.

ARGUMENTS
FMTX,FMTY (TYPE CHARACTER)
Format specifications for the X-axis and Y-axis numerical labels in GRIDL, PERIML, GRIDAL, or HALFX. The specification must start with a left parenthesis and end with a right parenthesis and should not use more than 8 characters. Only floating-point conversions (F, E, and G) such as FMTX='(F8.2)' and FMTY='(E10.0)' for example.

NUMX,NUMY (integer)
The number of characters specified by FMTX and FMTY. For the above examples, these would be NUMX=8 and NUMY=10 (not 6 and 7).

ISIZX,ISIZY
Character size codes for the labels. These size codes are the same as those for the SPPS entry PWRIT.
IXDEC
The decrement in plotter address units from the leftmost plotter coordinate (as specified by the current viewport) to the nearest X-address of the label specified by FMTY, NUMY, and ISIZY. For example, if the minimum X-coordinate of the current viewport is 102 (102*1024). If IXDEC is 60, the label will start at 42 (102-60). The following conventions are used:

- If IXDEC=0, it is automatically reset to properly position the Y-axis labels to the left of the left Y-axis, IXDEC=20.
- If IXDEC=1, Y-axis labels will go to the right of the graph, IXDEC=-20.

When either HALFAX or GRIDAL is called to draw an axis, IXDEC is the distance from the axis rather than from the minimum viewport coordinate.

IYDEC
The decrement in plotter address units from the minimum Y-axis coordinate as specified by the current viewport to the nearest Y-address of the label specified by FMTX, NUMX, and ISIZX. For example, if the minimum Y-coordinate of the current viewport is 205 (205*1024). If IYDEC=30, the label will end at 205-30=175. The following conventions are used:

- If IYDEC=0, it is automatically reset to properly position X-axis labels along the bottom, IYDEC=20.
- If IYDEC=1, X-axis labels will go along the top of the graph, IYDEC=-20.

IXOR (integer)
Orientation of the X-axis labels.

\[
\text{IXOR} = \begin{cases} 
0 & \text{+X (horizontal)} \\
1 & \text{+Y (vertical)} 
\end{cases}
\]

In normal orientation, the actual number of non-blank digits is centered under the line or tick to which it applies.

---------------
SUBROUTINE PERIM
---------------

April, 1986 5 GRIDAL
PURPOSE To draw a perimeter with tick marks.

USAGE CALL PERIM (MAJRX,MINRX,MAJRY,MINRY)

DESCRIPTION This subroutine behaves just as GRID except that interior lines are replaced with tick marks along the edges. Tick marks at major divisions are slightly larger than tick marks at minor divisions.

SUBROUTINE PERIML

PURPOSE To draw a perimeter with tick marks and labels.

USAGE CALL PERIML (MAJRX,MINRX,MAJRY,MINRY)

DESCRIPTION This subroutine behaves just as PERIM, but each major division is labeled with its numerical value.

SUBROUTINE TICK4

PURPOSE To allow program control of tick mark length.

USAGE CALL TICK4 (LMAJX,LMINX,LMAJY,LMINY)

DESCRIPTION This subroutine allows program control of tick mark length in PERIM, PERIML, GRIDAL, and HALFA.

ARGUMENTS

LMAJX, LMAJY
Length in plotter address units of major division tick marks on the X-axis and Y-axis respectively. These values are initially set to 12.

MINRX, MINRY
Length in plotter address units of minor division tick marks on the X-axis and Y-axis respectively. These values are initially set to 8.

We now resume the package documentation.

ENTRY POINTS GRID,GRIDAL,GRIDL,HALFA,LABMOD,PERIM,PERIML,TICK4, TICKS,CHSTR,EXPAND,GRIDT

COMMON BLOCKS LAB,CLAB,TICK,GRIINT

April, 1986
<table>
<thead>
<tr>
<th>REQUIRED ROUTINES</th>
<th>The ERPRT77 package and the SPPS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>Plots backgrounds</td>
</tr>
<tr>
<td>PRECISION</td>
<td>Single</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>FORTRAN 77</td>
</tr>
<tr>
<td>HISTORY</td>
<td>Written in June, 1984. Based on the NCAR System Plot Package entries having the same names.</td>
</tr>
</tbody>
</table>
HAFTON
SUBROUTINE HAFTON (Z,L,M,N,FLO,H1,NLEV,NOPT,NPRM,ISPV,SPVAL)

SUBROUTINE HAFTON (Z,L,M,N,FLO,H1,NLEV,NOPT,NPRM,ISPV,SPVAL)

DIMENSION OF ARGUMENTS

Z(L,M)

LATEST REVISION

July, 1984

PURPOSE

HAFTON draws a half-tone 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,H1,NLEV,
NOPT,NPRM,ISPV,SPVAL)

ARGUMENTS

ON INPUT

FOR EZHFTN

Z

M by N array to be used to generate a half-tone plot.

M

First dimension of Z.

N

April, 1986

1

HAFTON
Second dimension of Z.

All arguments are unchanged.

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 $|\text{ABS}(\text{NOPT})|$

- 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 half-tone 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 half-tone 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
All arguments are unchanged.

FOR HAFTON
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
GRIDAL, the ERPRT77 package and the SPPS.

ROUTINES
Plots half-tone picture.

I/O
Single

PRECISION
FORTRAN

LANGUAGE

HISTORY
Rewrite of PHOMAP originally written by M. Perry of High Altitude Observatory,

April, 1986
ALGORITHM
Bi-linear interpolation on plotter
(resolution-limited) grid of normalized
representation of data.

PORTABILITY
ANSI FORTRAN 77.

INTERNAL PARAMETERS
VALUES SET IN BLOCK DATA

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEFAULT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<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>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>SIDE</td>
<td>0.8</td>
<td>Length of longer edge of plot (see also EXT).</td>
</tr>
<tr>
<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>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 non-linear 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>MXLEV</td>
<td>16</td>
<td>Maximum number of levels. Limited by plotter.</td>
</tr>
<tr>
<td>NCRTG</td>
<td>8</td>
<td>Number of CRT units per gray-scale cell. Limited by plotter.</td>
</tr>
<tr>
<td>NCRTF</td>
<td>1024</td>
<td>Number of plotter address units per frame.</td>
</tr>
<tr>
<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></td>
<td>Intensities Used</td>
</tr>
<tr>
<td>2</td>
<td>5,11,</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4, 8,12,</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3, 6,10,13,</td>
<td></td>
</tr>
</tbody>
</table>

HAFTON

April, 1986

NCAR.
5 2, 5, 8, 11, 14,
6 1, 4, 7, 9, 12, 15,
7 1, 4, 6, 8, 10, 12, 15,
8 1, 3, 5, 7, 9, 11, 13, 15,
9 1, 3, 4, 6, 8, 10, 12, 13, 15,
10 1, 3, 4, 6, 7, 9, 10, 12, 13, 15,
11 1, 2, 3, 5, 6, 8, 10, 11, 13, 14, 15,
12 1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15,
13 1, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15,
14 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15,
15 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

April, 1986

HAFTON
HSTGRM
SUBROUTINE HSTGRM(DATA,NPTS,LOW,HI,NCLASS,WORK,LWORK,-IWORK,LIWORK)

PURPOSE
This routine plots a shaded histogram. The class intervals are labeled, a frequency axis is drawn on the left, and a per cent axis on the right. Lines are drawn through rectangles corresponding to the frequency-axis ticks. Plot options such as windowing, frame advance, a title, horizontal histogram, colors, etc., can be invoked by calling the routines HSTOPL, HSTOPR, and HSTOPC before calling HSTGRM.

USAGE
To draw a simple histogram (one with only the default features):

CALL HSTGRM (DATA,NPTS,LOW,HI,NCLASS,WORK,LWORK,LIWORK)

Note: Error handling is performed by the ERPRT77 centralized error-handling package. To recover from a warning (recoverable error), call ENSTR(IDUM,1) before calling HSTGRM, otherwise an error message is printed and the run is terminated.

ARGUMENTS

ON INPUT

DATA Real array containing data points, dimensioned NPTS.

NPTS Dimension of DATA array.

LOW Lower bound for data. All points .LT. LOW are plotted in the first histogram rectangle. These values are included when the median is computed.

HI Upper bound for data. All points .GE. HI are plotted in the last histogram rectangle. These values are included when the median is computed.

NCLASS Number of class intervals. Each interval = (HI - LOW)/NCLASS. A data point is in interval x if its value is .GE. interval x and .LT. interval x+1.

April, 1986
Note: The total number of class intervals is \( NCLASS + 2 \), one interval for points \( \text{LT. LOW} \) and one interval for points \( \text{GE. HI} \). \( NCLASS \) must be \( \geq 1 \).

WORK Real scratch array, dimensioned LWORK.

LWORK Dimension of the array WORK.
LWORK = \( 8 \times (NCLASS+3) + NPTS \) unless the shading option is turned off, then LWORK = \( 4 \times (NCLASS+3) + NPTS \).

IWORK Integer scratch array, dimensioned LIWORK.

LIWORK Dimension of IWORK.
LIWORK = \( 6 \times (NCLASS+3) \) unless the shading option is turned off, then LIWORK = 0.

ON OUTPUT All arguments remain unchanged except the scratch arrays WORK and IWORK which are written into.

ENTRY POINTS HSTBKD, HSTEXP, HSTMED, HSTOPC, HSTOPL, HSTOPR, HSTSTR, HSTGRM, HSTL, HSTLST, NWTSTR

COMMON BLOCKS HSTGC1, HSTGC1

I/O Plots histograms using various options.

PRECISION Single

REQUIRED The ERPRT77 package and the SPPS.

ADDITIONAL Routines

LANGUAGE FORTRAN

HISTORY First written in the fall of 1984.

ALGORITHM Determines the location of histograms and labels and draws the histograms.

PORTABILITY FORTRAN 77

We now describe the option-setting entries of the package.

------------------------------------------------------------

SUBROUTINE HSTOPL

April, 1986 2 HSTGRM
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 by 'OF'. Imbedded spaces are not allowed.

Example: CALL HSTOPL('PER=OFF')

The following options are turned 'ON' or 'OFF' by this routine (all defaults are listed at the end of the package documentation):

'HOR', horizontal
The histogram is drawn horizontally if 'HOR=ON'. If OFF, it is drawn vertically.

'PER', percent axis
If ON, a labeled percent axis is drawn on the right side of the histogram (or on top if horizontal.) If OFF, the percent axis is not drawn.

'MID', midvalues
If ON, the class interval labels are put at the midpoint of each interval. If off, all intervals are labeled.

'SHA', shading
If ON, the histogram rectangles are shaded. If OFF, the intervals are not shaded.

'MED', median
If ON, a line is drawn through the median of all points. If OFF, this line is not drawn.

'PRM', perimeter
If ON, a perimeter is drawn around the histogram. If OFF, no perimeter is drawn.

'FRA', frame
If ON, the frame is advanced automatically after the histogram is drawn. If OFF, the frame is not advanced, and the user must call FRAME.

April, 1986
'LIS', list options
   If ON, all the options along with their values are printed on the standard output.
   If OFF, nothing is printed on the standard output.

'DEF', default
   If ON, all the options are set to their default values; see list of default values below.

SUBROUTINE HSTOPR

PURPOSE To specify various REAL arrays to be used by the HSTGRM package.

USAGE CALL HSTOPR(STRING,ARRAY,LARR)

STRING A character string specifying the option to be set. The current possibilities are 'WIN' for setting the window, and 'COL' for setting the color.

ARRAY A real array of length LARR.

LARR Dimension of ARRAY.

EXAMPLE: REAL WINDOW(4)
   DATA WINDOW/.3,.7,.3,.7/
   CALL HSTOPR('WIN',WINDOW,4)

   The following arrays may be defined by this routine:

'COL', color options
   LARR = 7
   ARRAY(1) = color used for shading rectangles
   ARRAY(2) = color used for rectangle outlines
   ARRAY(3) = color used for drawing axes
   ARRAY(4) = color used for drawing median line
   ARRAY(5) = color used for text output (labels)
   ARRAY(6) = color used for title
   ARRAY(7) = color used for drawing perimeter

   The default is color index is 1 for all.

'WIN', Windowing
   LARR = 4
   ARRAY(1) = XMIN

April, 1986

HSTGRM
SUBROUTINE HSTOPC

PURPOSE To specify various CHARACTER variables to be used by the HSTGRM package.

USAGE CALL HSTOPC(STRING,STRNG2)

STRING A character string specifying which option is to be set.

STRNG2 A character string up to 45 characters long.

EXAMPLE: CALL HSTOPC('TIT','MY HISTOGRAM')

The following options are defined by this subroutine.

'FOR', format for numeric labels.
Default is '(G10.3)'

'TIT', title
Default is no title

'LAB', label for class intervals
Default is 'class intervals' (or 'class midvalues' if midvalue option is turned on).

'FQN', frequency label
Default is 'frequency'

The defaults for all the options are:

HOR = OFF
PER = ON
MID = OFF
SHA = ON
MED = OFF
PRM = OFF
FRA = ON
LIS = OFF
WIN = OFF, maximum viewport is used
COL = OFF, color index 1 is used for all output

Assumptions:
XMIN .LT. XMAX and YMIN .LT. YMAX .
XMIN, XMAX, YMIN, and YMAX must be in the range 0. to 1.
FOR = OFF, format = '(G10.3)'
TIT = OFF, no title
LAB = OFF, label = 'class intervals' or 'class midvalues'
FQN = OFF, frequency label = 'frequency'
ISOSRF
SUBROUTINE ISOSRF (T,LU,MU,LV,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)

DIMENSION OF T(LU,LV,MW),EYE(3),SLAB(MUVWP2,MUVWP2)

LATEST REVISION December 1984

PURPOSE ISOSRF draws an approximation of an iso-valued surface from a three-dimensional array with hidden lines removed.

USAGE If the following assumptions are met, use

CALL EZISOS (T,MU,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,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)

ARGUMENTS

ON INPUT

T
  Three dimensional array of data that defines the iso-valued 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).

April, 1986 1 ISOSRF
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*MU,3.5*MV,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.

- 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>

- Second, the sign of IFLAG determines what is inside and what is outside, hence, which lines are visible and what is done

April, 1986 ISOSRF
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.

**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, PWRZI

**REQUIRED LIBRARY ROUTINES**

The ERPRT77 package and the SPPS.

**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 which also marks a model of the plotting plane. Interiors of boundaries are filled in and the result is

April, 1986
.OR.ed into another model of the plotting plane which is used to test subsequent contour lines for visibility.

**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>Flag to control drawing of axes. .IREF=nonzero Draw axes. .IREF=zero Do not draw axes.</td>
<td></td>
</tr>
</tbody>
</table>

April, 1986 4 ISOSRF
ISOSRFHR
SUBROUTINE INIT3D (EYE,NU,NV,NW,ST1,LX, NY, IS2, IU, S)

ISO-SURFACE PACKAGE FOR HIGH RESOLUTION 3-DIMENSIONAL ARRAYS

DIMENSION OF EYE(3), ST1(NV,NW,2), IS2(LX, NY), S(4),
             IOBJS(MV,NW)

LATEST REVISION July 1984

PURPOSE

This package of three routines produces a
perspective picture of an arbitrary object or
set of objects with the hidden parts not
drawn. The objects are assumed to be stored
in the format described below, a format which
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
equivalent follows:

CALL INIT3D(EYE, NU, NV, NW, ST1, LX, NY,
             IS2, IU, S)
DO 1 IBKWD = 1, NU
   I = NU+1-IBKWD
C   FORM OR READ SLAB I OF THE 3 DIMENSIONAL

April, 1986

1

ISOSRFHR
C 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 possible).
need be). As a guide, some examples showing successful choices are listed:

<table>
<thead>
<tr>
<th>GIVEN NU NV NW</th>
<th>RESULTING NX NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 100 60</td>
<td>200 200</td>
</tr>
<tr>
<td>60 60 60</td>
<td>110 110</td>
</tr>
<tr>
<td>40 40 40</td>
<td>75 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 $S(2)-S(1) = S(4)-S(3)$. $10.0 \leq S(I) \leq 100.0$ for $I = 1,2,3,4$.

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 sited below.


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.

April, 1986
SUBROUTINE PWRITX(X,Y,IDPC,NCHAR,JSIZE,JOR,JCTR)

SUBROUTINE PWRITX (X,Y,IDPC,NCHAR,JSIZE,JOR,JCTR)

LATEST REVISION August, 1984

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 sample plots section of the current graphics 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

April, 1986 1 PWRITX
CALL PWRITX(................

.
.
.

NOTES
The character set cannot be changed after
the first call to PWRITX. To produce
examples of the duplex character set,
run the demo driver for pwrîtčx (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
X,Y
Positioning coordinates for the characters to
be drawn. These are world coordinates. See
JCTR.

IDPC
Characters to be drawn and FUNCTION CODES
(see below.) IDPC is TYPE CHARACTER.

NCHAR
The number of characters in IDPC, including
characters to be drawn and function codes.

JSIZE
Size of the character.
. If between 0 and 3, it is 1., 1.5, 2.,
or 3. times digitized character width.
(See -FUNCTION CODES- below for these
sizes.)
. If greater than 3 it is the desired plotter
address units for principal character
height, i.e. 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 h

JOR
Character string orientation in degrees
counter-clockwise from the positive X axis.

April, 1986

PWRITX
ON OUTPUT
FUNCTION CODES

JCTR
  Centering option.
  = 0 (X,Y) is the center of the entire string.
  = -1 (X,Y) is the center of the left edge of the first character.
  = 1 (X,Y) is the center of the right edge of the last character.

All arguments are unchanged.

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)

A. FONT DEFINITIONS
  R Roman type characters
  G Greek type characters

B. 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...
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.

C. 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.

D. 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: 'Ul'T'Sl'EST
          E
          Will be written TST
'C
          'Ul'T'S'E'N'ST
          E
          Will be written T ST

E. 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.

F. 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.

G. DIRECT CHARACTER ACCESS
NNN. 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 last PWRITX call.
. 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 RESIDENT Routines
SQR, SIN, COS.

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

April, 1986 6
REQUIRED LIBRARY
The ERPRT77 package and the SPPS.

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

<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 chars.</td>
</tr>
<tr>
<td>SINDW</td>
<td>12.</td>
<td>Width of indexical chars.</td>
</tr>
<tr>
<td>SCARH</td>
<td>14.</td>
<td>Height of cartographic chars.</td>
</tr>
<tr>
<td>SCARW</td>
<td>8.</td>
<td>Width of cartographic chars.</td>
</tr>
<tr>
<td>SSPR</td>
<td>10.</td>
<td>Shift in case of super or subscripting for principal characters.</td>
</tr>
<tr>
<td>SSIC</td>
<td>7.</td>
<td>Shift in case of super or subscripting for indexical characters.</td>
</tr>
</tbody>
</table>

All values below are for plotter address units.

IMPLEMENTATION INSTRUCTIONS

1. Create the PWRTX binary database from the card-image files PWRTXC1, PWRTXC2, PWRTXD1, PWRTXD2 supplied. To do this, read the instructions in file PWRTXINIT and execute that program.

2. Supply 3 machine dependent subprograms for bit manipulations, namely ISHIFT(I1,I2), IOR(I1,I2), and IAND(I1,I2). These subprograms can be the same routines used to implement the NCAR GKS plot package.

3. Set up a binary input file from which character data may be read (user input) and supply its number below.

4. Set the binary input file where PWRTX may read its data.
PWRITY
SUBROUTINE PWRITY (X,Y,ID,N,ISIZE,ITHETA,ICNT)

LATEST REVISION
July 1984

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 (counter clockwise 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 od the right edge of the last character.

ON OUTPUT
All arguments are unchanged.

ENTRY POINTS
PWRY, PWRYSO, PWRYGT, PWRITY, PWRYBD

April, 1986
PWRITY
COMMON BLOCKS PWRCOM
REQUIRED LIBRARY The SPPS.
PWRZI
SUBROUTINE PWRZI (X,Y,Z,ID,N,IISIZE,LIN3,ITOP,ICNT)

LATEST REVISION July, 1984

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,IISIZE,LINE,ITOP,ICNT)
Use 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.

IISIZE Size of the character:
  If between 0 and 3, IISIZE is 1., 1.5, 2., OR 3. times a standard width equal to 1/128th of the screen width.
  If greater than 3, IISIZE 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

April, 1986
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, PWRZOI, PWRZGI

COMMON BLOCKS PWRZII, PWRZ2I

I/O Plots character(s)

PRECISION Single

REQUIRED LIBRARY Routines ISOSRF, the ERPRT77 package, and the SPPS

LANGUAGE FORTRAN

HISTORY Implemented for use with ISOSRF.
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.

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.

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
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 SRFACE, the ERPRT77 package, and the SPPS ROUTINES
LANGUAGE FORTRAN
HISTORY Implemented for use with SRFACE.

******************************************
PWRZT
SUBROUTINE PWRZT (X,Y,Z,ID,N,ISIZE,LIN3,ITOP,ICNT)

LATEST REVISION
July, 1984

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,LINE,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 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
PWRZT, INITZT, PWRZOT, PWRZGT

COMMON BLOCKS
PWRZ1T, PWRZ2T

I/O
Plots character(s)

PRECISION
Single

REQUIRED LIBRARY
THREED, the ERPRT77 package, and the SPPS ROUTINES

LANGUAGE
FORTRAN

HISTORY
Implemented for use with THREED.

*****************************************************************************
SRFACE
SUBROUTINE SRFACE (X,Y,Z,M,MX,NX,NY,S,STEREO)

DIMENSION OF X(NX),Y(NY),Z(MX, NY),M(2,NX,NY),S(6)

ARGUMENTS

LATEST REVISION March 1984

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 (counter-clockwise 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

April, 1986 1 SRFACE
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 note, below.

Y
The linear array NY long containing the Y coordinates of the points in the surface approximation. See note, 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 note, 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
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
June 1984

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 (i.e., stereographic), then the appropriate transformation must be made to the U and V components via the functions FU and FV (located in DRWSTR).)

April, 1986
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)

JPTSY
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 |
| HISTORY | Written and standardized in November 1973. Draws streamlines |
| I/O | Single |
| PRECISION | FORTRAN |
| 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 Whittaker (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; |

April, 1986
a fairly complex flow field will take about 1.5 seconds on the CRAY1-A.

INTERNAL PARAMETERS

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEFAULT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<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,JPTY) / MAX(IPTSX,JPTSY) .LT. EXT; in that case a square graph is plotted.</td>
</tr>
<tr>
<td>SIDE</td>
<td>0.90</td>
<td>Length of longer edge of plot. (See also EXT.)</td>
</tr>
<tr>
<td>XLT</td>
<td>0.05</td>
<td>Left hand edge of the plot. (0.0 = left edge of frame) (1.0 = right edge of frame)</td>
</tr>
<tr>
<td>YBT</td>
<td>0.05</td>
<td>Bottom edge of the plot. (0.0 = bottom ; 1.0 = top) (YBT+SIDE and XLT+SIDE must be .LE. 1.)</td>
</tr>
<tr>
<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>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>
</tbody>
</table>

April, 1986
AROWL 0.33 Length of direction arrow. For example, 0.33 means each directional arrow will take up a third of a grid box.

ITERP 35 Every 'ITERP' iterations the streamline progress is checked.

ITERC -99 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.) Caution: When this parameter is activated CPU time will increase.

IGFLG 0 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.

IMSG 0 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.

UVMSG  1.E+36  Value assigned to a missing point.

ICYC  0  Zero means the data are non-cyclic in the X direction.
If .NE 0, the cyclic interpolation formulas will be used.
(Note: Even if the data are cyclic in X leaving ICYC = 0 will do no harm.)

DISPL  0.33  The wind speed is normalized to this value.
(See the discussion below.)

DISPC  0.67  The critical displacement.
If after 'ITERP' iterations the streamline has not moved this distance, the streamline will be terminated.

CSTOP  0.50  This parameter controls the spacing between streamlines. The checking is done when a new grid box is entered.

DISCUSSION OF DISPL,DISP C 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 streamlines will be drawn per grid box. This max will normally only occur in areas of singular points.

************************************
Any or all of the above parameters may be changed.

April, 1986  6  STRMLN
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.

April, 1986 7 STRMLN
THREED
SUBROUTINE SET3 (XA, XB, YA, YB, ULO, UHI, VLO, VHI, WLO, WHI, EYE)

THREE-DIMENSIONAL LINE DRAWING PACKAGE

LATEST REVISION July, 1984

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 INWHICH 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 CURVE, 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.

All arguments are unchanged.

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
PWRZ and the SPPS

HISTORY
Written and standardized in November 1973. Plots lines.

I/O
Single

PRECISION
FORTRAN

ACCURACY
+ or -.5 plotter address units per call. There is no cumulative error.

PORTABILITY
ANSI FORTRAN 77

April, 1986
VELVCT
SUBROUTINE VELVCT (U,LU,V,LV,M,N,FLO,HI,NSET,LENGTH,ISPV,SPV)

DIMENSION OF U(LU,N),V(LV,N),SPV(2)

LATEST REVISION July 1984

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
SQT(U(I,J)**2+V(I,J)**2) and direction
ATAN2(V(I,J),U(I,J)). Other representations,
such as (R,THETA), can be plotted by
changing statement functions in this routine.

LU

The first dimension of U in the calling

April, 1986
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 \( \sqrt{U^2 + V^2} \) 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 \( U(I,J)=SPV(1) \) the vector will not be plotted.

2 means that if the value of \( V(I,J)=SPV(2) \) the vector will not be plotted.

3 means that if either \( U(I,J)=SPV(1) \) or \( V(I,J)=SPV(2) \) then the vector will not be plotted.

4 means that if \( U(I,J)=SPV(1) \) and \( V(I,J)=SPV(2) \), the vector will not be plotted.

SPV

An array of length 2 which gives the value in the U array and the value in the V array which 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 \((MXF(X,Y,U,V,SFX,SYF,MY,MY))\), \(MYF(X,Y,U,V,SFX,SYF,MY,MY))\) 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.

April, 1986 3 VELVCT
Thus the actual length of the arrow is $\sqrt{DX^2+DY^2}$ 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

**REQUIRED LIBRARY ROUTINES**
GRIDAL and the SPPS

**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 FORTRAN77 and GKS in July 1984.

**ALGORITHM**
Each vector is examined, possibly transformed, then plotted.

**PORTABILITY**
FORTRAN77

---

**SPECIAL 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) *10 degrees east of Greenwich and a latitude (J-1)*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)

April, 1986
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.

FUNCTION FX(XX, YY)
FX = X
RETURN
END

FUNCTION FY(XX, YY)
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 basepoints 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.

DECLARATIONS -
EXAMPLES
## CONTENTS

<table>
<thead>
<tr>
<th>Utility Plots</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOGRAPH</td>
<td>2</td>
</tr>
<tr>
<td>CONRAN</td>
<td>4</td>
</tr>
<tr>
<td>CONRAQ</td>
<td>6</td>
</tr>
<tr>
<td>CONRAS</td>
<td>8</td>
</tr>
<tr>
<td>CONREC</td>
<td>10</td>
</tr>
<tr>
<td>CONRECQOCK</td>
<td>12</td>
</tr>
<tr>
<td>CONRECSMTH</td>
<td>14</td>
</tr>
<tr>
<td>CONRECSSUPR</td>
<td>16</td>
</tr>
<tr>
<td>DASHCHAR</td>
<td>18</td>
</tr>
<tr>
<td>DASHLINE</td>
<td>22</td>
</tr>
<tr>
<td>DASHSMTH</td>
<td>26</td>
</tr>
<tr>
<td>DASHSUPR</td>
<td>30</td>
</tr>
<tr>
<td>EZMAP</td>
<td>34</td>
</tr>
<tr>
<td>GRIDAL</td>
<td>48</td>
</tr>
<tr>
<td>HAFTON</td>
<td>52</td>
</tr>
<tr>
<td>HSTGRM</td>
<td>54</td>
</tr>
<tr>
<td>ISOSRF</td>
<td>56</td>
</tr>
<tr>
<td>ISOSSRFHR</td>
<td>58</td>
</tr>
<tr>
<td>PWRITX</td>
<td>62</td>
</tr>
<tr>
<td>PWRITY</td>
<td>70</td>
</tr>
<tr>
<td>PWRZI</td>
<td>72</td>
</tr>
<tr>
<td>PWRZS</td>
<td>74</td>
</tr>
<tr>
<td>PWRZT</td>
<td>76</td>
</tr>
<tr>
<td>SRFACE</td>
<td>78</td>
</tr>
<tr>
<td>STRMLN</td>
<td>80</td>
</tr>
<tr>
<td>THREEED</td>
<td>82</td>
</tr>
<tr>
<td>VELVCT</td>
<td>84</td>
</tr>
<tr>
<td>Non-trivial plots</td>
<td>87</td>
</tr>
<tr>
<td>Code</td>
<td>89</td>
</tr>
<tr>
<td>AUTOGRAPH</td>
<td>95</td>
</tr>
<tr>
<td>CONREC with polar transformation</td>
<td>96</td>
</tr>
<tr>
<td>CONREC on EZMAP</td>
<td>97</td>
</tr>
<tr>
<td>CONREC on EZMAP</td>
<td>98</td>
</tr>
<tr>
<td>CONREC</td>
<td>99</td>
</tr>
<tr>
<td>VELVCT on EZMAP</td>
<td>100</td>
</tr>
<tr>
<td>Example of GKS output primitives</td>
<td>101</td>
</tr>
<tr>
<td>Code</td>
<td>103</td>
</tr>
<tr>
<td>Plot</td>
<td>105</td>
</tr>
<tr>
<td>Simple plot with hex and octal dumps of metafile</td>
<td>107</td>
</tr>
<tr>
<td>Code</td>
<td>109</td>
</tr>
<tr>
<td>Plot</td>
<td>113</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>SPPS test</td>
<td>115</td>
</tr>
<tr>
<td>Code</td>
<td>117</td>
</tr>
<tr>
<td>Plots</td>
<td></td>
</tr>
<tr>
<td>PWRIT</td>
<td>126</td>
</tr>
<tr>
<td>PLOTIT</td>
<td>127</td>
</tr>
<tr>
<td>LINE</td>
<td>128</td>
</tr>
<tr>
<td>CURVE</td>
<td>129</td>
</tr>
<tr>
<td>VECTOR</td>
<td>130</td>
</tr>
<tr>
<td>SET</td>
<td>131</td>
</tr>
<tr>
<td>PERIM</td>
<td>132</td>
</tr>
<tr>
<td>HALFAX</td>
<td>133</td>
</tr>
<tr>
<td>TICK4</td>
<td>134</td>
</tr>
<tr>
<td>GRID</td>
<td>135</td>
</tr>
<tr>
<td>GRIDL</td>
<td>136</td>
</tr>
<tr>
<td>PERIML</td>
<td>137</td>
</tr>
<tr>
<td>LABMOD</td>
<td>138</td>
</tr>
<tr>
<td>POINT</td>
<td>139</td>
</tr>
<tr>
<td>POINTS</td>
<td>140</td>
</tr>
<tr>
<td>GRIDAL</td>
<td>141</td>
</tr>
</tbody>
</table>
In this section we present a simple example of each of the Graphics Utilities. The example is in the form of a small program. Generally the program fits on a single page and the plot is on the facing page. A few of the programs could not be fit onto one page, in these cases the plot will be on a separate page so that the code and the plot may be examined simultaneously. A few of the Graphics Utilities produce more than one plot because they have many distinct applications.

In cases where a Graphics Utility has an easy version, the code for the easy version is included as comments in the example program. The plot produced for these Utilities is from the full calling sequence.

Each sample program contains in the comments a list of other Utilities that must be present in order for the program to execute. In all cases the System Plot Package Simulator (SPPS) must be present, as well as a GKS package.
PROGRAM TAUTOG

C PURPOSE PROVIDES A DEMONSTRATION OF AUTOGRAPH
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE SAMPLE PLOT IS PRODUCED.
C REQUIRED ROUTINES DASHCHAR
C
REAL X(21), Y2D(21,5)
C X CONTAINS THE ABSCISSA VALUES FOR THE PLOT PRODUCED BY EZMXY. AND Y2D CONTAINS THE ORDINATE VALUES FOR THAT PLOT.
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C FILL X ARRAY
DO 20 I=1,21
  X(I) = FLOAT(I-1)*.314
20 CONTINUE
C FILL Y2D ARRAY FOR EZMXY
DO 60 I=1,21
  DO 50 J=1,5
    Y2D(I,J) = X(I)**J+COS(X(I))
  50 CONTINUE
60 CONTINUE
C SET AUTOGRAPH CONTROL PARAMETERS FOR Y-AXIS LABEL
CALL AGSETC('LABEL/NAME.','L')
CALL AGSETI('LINE/NUMBER.','100')
CALL AGSETC('LINE/TEXT.','X**J+COS(X)$')
C SET AUTOGRAPH CONTROL PARAMETER FOR SPECIFYING THAT THE ALPHABETIC SET OF DASHED LINE PATTERNS IS TO BE USED.
CALL AGSETI('DASH/SELECTOR.','-1')
C SET AUTOGRAPH CONTROL PARAMETER FOR SPECIFYING THAT THE GRAPH IS TO BE LINEAR IN THE X-AXIS AND LOGARITHMIC IN THE Y-AXIS.
CALL AGSETI('X/LOGARITHMIC.',0)
CALL AGSETI('Y/LOGARITHMIC.',1)
C ENTRY EZMXY PLOTS MULTIPLE ARRAYS AS A FUNCTION OF A SINGLE X ARRAY (OR MANY X ARRAYS)
CALL EZMXY (X,Y2D,21,5,21,
  + 'DEMONSTRATING EZMXY ENTRY OF AUTOGRAPH$')
WRITE (6,1001)
1001 FORMAT (' AUTOGRAPH TEST SUCCESSFUL',24X,
  'SEE PLOT TO VERIFY PERFORMANCE')
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END

April 1986

-2-
PROGRAM TCONAN

C PURPOSE
PROVIDES A SIMPLE DEMONSTRATION OF CONRAN
THIS SAME SUBROUTINE CAN BE USED TO PRODUCE
DEMO PLOTS OF THE SMOOTH VERSION OF CONRAN
BY LOADING DASHSMTH INSTEAD OF DASHCHAR.

C I/O
IF THE TEST IS SUCCESSFUL, A MESSAGE IS
PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
CONTAINING THE CONTOUR PLOT IS PRODUCED.
A SPARSE DATA SET IS DEFINED VIA DATA
STATEMENTS AND OPTIONS SET TO PRODUCE A TITLE.
MESSAGE AND PERIMETER ARE PRODUCED BY DEFAULT.

C REQUIRED ROUTINES
CONTERP, CONCOM, DASHCHAR OR DASHSMTH, GRIDAL

C SET UP THE SCRATCH SPACES REQUIRED BY CONRAN
DIMENSION WK(221),IWK(744),SCR(1600)

C SET UP AND FILL THE ARRAYS TO DEFINE THE DATA SET
DIMENSION XD(17),YD(17),ZD(17)
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., 18., 10., 10., 5., 1.,
  3 15., 20., 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., 18., 18., 3., 18., 10., 1., 5., 10.,
  3 5., 10., 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),
  3 1., 1., 1., 1., 1., 1., 1., 25./
DATA NDP/17/

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C SET UP TITLE FOR THE PLOT
CALL CONOP4('TLE=ON','DEMONSTRATION PLOT FOR CONRAN',29,0)

C CALL CONRAN TO CONTOUR DATA AND CALL FRAME
CALL CONRAN(XD,YD,ZD,NDP,WK,IWK,SCR)
CALL FRAME

C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS

C PRINT MESSAGE EVERYTHING OK
WRITE(6,10)
10 FORMAT(1X, 'CONRAN SUCCESSFUL, SEE PLOT TO CHECK PERFORMANCE')
STOP
END
DEMONSTRATION PLOT FOR CONRAN

CONTOUR FROM -4.0000 TO 26.000 CONTOUR INTERVAL OF 2.0000
1.9000 Y INTERVAL= 1.9000
April 1986
PROGRAM TCONAQ

C PURPOSE
  PROVIDES A SIMPLE DEMONSTRATION OF CONRAQ

C I/O
  IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE CONTOUR PLOT IS PRODUCED.
  A SPARSE DATA SET IS DEFINED VIA DATA STATEMENTS AND OPTIONS SET TO PRODUCE A TITLE, MESSAGE AND PERIMETER ARE PRODUCED BY DEFAULT.

C REQUIRED ROUTINES
  CONTERP, CONCOM, DASHLINE, GRIDAL

C SET UP THE SCRATCH SPACES REQUIRED BY CONRAQ
  DIMENSION WK(221),IWK(744)

C SET UP AND FILL THE ARRAYS TO DEFINE THE DATA SET
  DIMENSION XD(17), YD(17), ZD(17)
  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., 18., 10., 10., 5., 1.,
  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., 10., 18., 18., 10., 10., 5., 1.,
  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),
  DATA NDP/17/

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
  CALL GOPKS(6)
  CALL GOPWK(1,2,1)
  CALL GACWK(1)

C SET UP TITLE FOR PLOT
  CALL CONOP4('TITLE=ON', 'DEMONSTRATION PLOT FOR CONRAQ', 29, 0)

C CALL CONRAQ TO CONTOUR DATA, CALL FRAME
  CALL CONRAQ(XD, YD, ZD, NDP, WK, IWK)
  CALL FRAME

C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS
  CALL GDAWK(1)
  CALL GCLWK(1)
  CALL GCLKS

C PRINT MESSAGE EVERYTHING OK
  WRITE(6,10)
  10 FORMAT('CONRAQ TEST SUCCESSFUL, SEE PLOT TO VERIFY', 1 ' PERFORMANCE')
  STOP
END

April 1986
-6-
DEMONSTRATION PLOT FOR CONRAQ

CONTOUR FROM -4.0000 TO 26.000 CONTOUR INTERVAL OF 2.0000
X INTERVAL= 1.9000 Y INTERVAL= 1.9000

April 1986 -7- EXAMPLES
PROGRAM TCONAS

C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF CONRAS

C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE CONTOUR PLOT IS PRODUCED.

C A SPARSE DATA SET IS DEFINED VIA DATA STATEMENTS AND OPTIONS SET TO PRODUCE A TITLE. MESSAGE AND PERIMETER ARE PRODUCED BY DEFAULT.

C REQUIRED ROUTINES CONTERP, CONCOM, DASHSUPR, GRIDAL, ENCD

C SET UP THE SCRATCH SPACES REQUIRED BY CONRAS

DIMENSION WK(221), IWK(744), SCR(1600)

C SET UP THE ARRAYS TO DEFINE THE DATA SET

DIMENSION XD(17), YD(17), ZD(17)

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., 18., 10., 10., 5., 1.,
3 15., 20., 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., 18., 18., 3., 18., 10., 1., 5., 10.,
3 5., 10., 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),
3 1., 1., 1., 1., 1., 1., 1., 25./

DATA NDP/17/

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION

CALL GOPKS(6)
CALL G0PWK(1, 2, 1)
CALL GACWK(1)

C SET UP TITLE FOR PLOT

CALL CONOP4('TLF=ON', 'DEMONSTRATION PLOT FOR CONRAS', 29, 0)

C CALL CONRAS TO CONTOUR DATA AND CALL FRAME

CALL CONRAS(XD, YD, ZD, NDP, WK, IWK, SCR)
CALL FRAME

C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS

CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS

C PRINT MESSAGE EVERYTHING OK

WRITE(6, 10)
10 FORMAT(' CONRAS TEST SUCCESSFUL, SEE PLOT TO VERIFY ',
1 'PERFORMANCE')
STOP
END

April 1986
PROGRAM TCONRE
C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF CONREC
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE CONTOUR PLOT IS PRODUCED.
C
C REQUIRED Routines GRIDAL, AND DASHCHAR
C
C Z CONTAINS THE VALUES TO BE PLOTTED.
REAL Z(21,25)
C
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM
C 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/.4267/, TY/.9765/
C
C FILL TWO DIMENSIONAL ARRAY TO BE PLOTTED
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 1./((X+.10)**2+Y**2+.09)
   10 CONTINUE
20 CONTINUE
C
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK()
C...
C SELECT NORMALIZATION TRANSFORMATION NUMBER 0
CALL GSELNT ( 0 )
C ENTRY EZCNTR REQUIRES ONLY THE ARRAY NAME AND ITS DIMENSIONS.
C FOR EXAMPLE:
C CALL EZCNTR (Z,21,25)
C
C ENTRY CONREC ALLOWS USER SPECIFICATION OF PLOT PARAMETERS. IN THIS
C EXAMPLE, THE LOWEST CONTOUR LEVEL (-4.5), THE HIGHEST CONTOUR LEVEL
C (4.5), AND THE INCREMENT BETWEEN CONTOUR LEVELS (0.3) ARE SPECIFIED.
C THE PLOT IS TITLED BEFORE CALLING CONREC.
C CALL WTSTR ( TX ,TY,
C 1 'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONREC',2,0,0 )
C CALL CONREC (Z,21,21,25,-4.5,4.5,.3,0,0,0)
C CALL FRAME
C
C WRITE (6,1001)
1001 FORMAT (' CONREC TEST SUCCESSFUL',24X,
C 1 'SEE PLOT TO VERIFY PERFORMANCE')
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END
April 1986
-10-
EXAMPLES
DEMONSTRATION PLOT FOR CONREC ENTRY OF CONREC

CONTOUR FROM -4.5000 TO 4.5000
CONTOUR INTERVAL OF 0.3000
PT(3,3) = -1.9066

April 1986

EXAMPLES-11-
PROGRAM TCNQCK

C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF CONRECQCK
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE CONTOUR PLOT IS PRODUCED.
C
C REQUIRED ROUTINES DASHLINE
C
C Z CONTAINS THE VALUES TO BE PLOTTED.
REAL Z(21,25)
C
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM
C 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/.4267/, TY/.9765/
C
C FILL TWO DIMENSIONAL ARRAY TO BE PLOTTED
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 OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C SELECT NORMALIZATION TRANSFORMATION NUMBER 0
CALL GSELNT(0)
C
C ENTRY EZCNTR REQUIRES ONLY THE ARRAY NAME AND ITS DIMENSIONS.
C FOR EXAMPLE:
C CALL EZCNTR (Z,21,25)
C
C ENTRY CONREC ALLOWS USER SPECIFICATION OF PLOT PARAMETERS. IN THIS
C EXAMPLE, THE LOWEST CONTOUR LEVEL (-4.5), THE HIGHEST CONTOUR LEVEL
C (4.5), AND THE INCREMENT BETWEEN CONTOUR LEVELS (0.3) ARE SPECIFIED.
C THE PLOT IS TITLED BEFORE CALLING CONREC.
CALL WTSTR ( TX ,TY, 1 : 'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECQCK',2,0,0 )
CALL CONREC (Z,21,21,21,25,-4.5,4.5,.3,0,0,0)
CALL FRAME
C
C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
C
WRITE (6,1001)
1001 FORMAT (' CONRECQCK TEST SUCCESSFUL',24X,
1 'SEE PLOT TO VERIFY PERFORMANCE')
STOP
END
DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECQCK

CONTOUR FROM -4.5000 TO 4.5000 CONTOUR INTERVAL OF 0.30000 PT(3.3)= -1.9066

April 1986
PROGRAM TCONRE

C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF CONRECSMTH
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE CONTOUR PLOT IS PRODUCED.
C
C REQUIRED ROUTINES GRIDAL, AND DASHSMTH
C
C Z CONTAINS THE VALUES TO BE PLOTTED.
REAL Z(21,25)
C
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/.4267/, TY/.9765/
C
C FILL TWO DIMENSIONAL ARRAY TO BE PLOTTED
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.
      1./((X+.10)**2+Y**2+.09)
   10 CONTINUE
20 CONTINUE
C
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK (1)
C
C SELECT NORMALIZATION TRANSFORMATION NUMBER 0
CALL GSELNT (0)
C
C ENTRY EZCNTR REQUIRES ONLY THE ARRAY NAME AND ITS DIMENSIONS.
C FOR EXAMPLE:
CALL EZCNTR (Z,21,25)
C
C ENTRY CONREC ALLOWS USER SPECIFICATION OF 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 SPECIFIED. THE PLOT IS TITLED BEFORE CALLING CONREC.
CALL WTSTR (TX ,TY, 'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSMTH',2,0,0)
CALL CONREC (Z,21,21,25,-4.5,4.5,.3,0,0,0)
CALL FRAME
WRITE (6,1001)
1001 FORMAT (' CONREC TEST SUCCESSFUL'.,24X, 'SEE PLOT TO VERIFY PERFORMANCE')
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END
DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSMTH

CONTOUR FROM -4.5000 TO 4.5000

CONTOUR INTERVAL OF 0.30000

PT(3,3) = -1.9066

April 1986

EXAMPLES
PROGRAM TCNSUP
C
C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF CONRECSUPR
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE CONTOUR PLOT IS PRODUCED.
C
C REQUIRED ROUTINES ENCD, GRIDAL, DASHSUPR
C
C Z CONTAINS THE VALUES TO BE PLOTTED.
REAL Z(21,25)
C
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM
C 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/.4267/, TY/.9765/
C
C FILL TWO DIMENSIONAL ARRAY TO BE PLOTTED
DO 20 I=1,21
  DO 10 J=1,25
    X = .1*FLOAT(I-11)
    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 OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C SELECT NORMALIZATION TRANSFORMATION NUMBER 0
CALL GSELNT(0)
C
C ENTRY EZCNTR REQUIRES ONLY THE ARRAY NAME AND ITS DIMENSIONS.
C FOR EXAMPLE:
C CALL EZCNTR (Z,21,25)
C
C ENTRY CONREC ALLOWS USER SPECIFICATION OF PLOT PARAMETERS, IF DESIRED.
C IN THIS EXAMPLE, THE LOWEST CONTOUR LEVEL (-4.5), THE HIGHEST CONTOUR
C LEVEL (4.5), AND THE INCREMENT BETWEEN CONTOUR LEVELS (0.3) ARE
C SPECIFIED. ALSO THE LABELLING OF THE HIGHS AND LOWS IS SUPPRESSED.
C
C THE TITLE FOR THIS PLOT IS:
C DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSUPR
CALL WTSTR(TX,TY,
  1 'DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSUPR',
  2 2,0,0)
CALL CONREC(Z,21,21,25,-4.5,4.5,.3,0,-1,0)
CALL FRAME
C
WRITE (6,1001)
1001 FORMAT (' CONRECSUPR TEST SUCCESSFUL',24X,
  1 'SEE PLOT TO VERIFY PERFORMANCE')
STOP
END
April 1986 -16- EXAMPLES
DEMONSTRATION PLOT FOR CONREC ENTRY OF CONRECSUPR

CONTOUR FROM -4.5000 TO 4.5000
CONTOUR INTERVAL OF 0.30000
PT(3,3) = -1.9066

March 1986

April 1986
PROGRAM TDASHC

C PURPOSE TO PROVIDE A DEMONSTRATION OF DASHCHAR

C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING FIVE CURVES EACH USING A DIFFERENT DASHCHAR PATTERNS IS PRODUCED.

C REQUIRED ROUTINES NONE

C X CONTAINS ABSCISSAE AND Y THE ORDINATE VALUES OF THE CURVE

DIMENSION X(31), Y(31)

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION

CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C SET SOLID DASH PATTERN, 1111111111111111 (BINARY).
C BOOLEAN OPERATIONS (EMPLOYING LOCALLY-IMPLEMENTED SUPPORT ROUTINES) ARE USED FOR PORTABILITY 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 DRAW CENTRAL AXIS FOR EACH CURVE

CALL FRSTD (.50, ORG-0.03)
CALL VECTD (.50, ORG+0.03)
CALL LINED (.109, ORG, .891, ORG)

C CALL SUBROUTINE DASHDC WITH DIFFERENT DASHED PATTERNS

DO 130 K=1,5
  CALL DASHDC ('$'K=1', 10, 12)
  CALL DASHDC ('$'K=2', 10, 12)
  CALL DASHDC ('$'K=3', 10, 12)
  CALL DASHDC ('$'K=4', 10, 12)
  CALL DASHDC ('$'K=5', 10, 12)
GO TO 60

C COMPUTE VALUES FOR AND DRAW THE KTH CURVE

DO 70 I=1,31
  THETA = FLOAT(I-1)*3.1415926535897932/15.
  X(I) = 0.5 + 0.4*COS(THETA)
  Y(I) = ORG + 0.075*SIN(FLOAT(K)*THETA)
70 CONTINUE

CALL CURVED (X, Y, 31)

C SET TEXT ALIGNMENT TO CENTER THE STRING AT THE LEFT OF THE STRING AND IN THE VERTICAL CENTER.

CALL GSTXAL(1,3)
CALL GSCHH(.012)

C LABEL EACH CURVE WITH THE APPROPRIATE CHARACTER REPRESENTATION.

ORY = ORG + .089
GO TO ( 80, 90, 100, 110, 120), K

80 CALL GTX(.1, ORY,'IPAT=DADADADADADADK=1')
90 CALL GTX(.1, ORY,'IPAT=DDDDDDADADDADDK=2')
100 CALL GTX(.1, ORY,'IPAT=DDDDDDDDDDDDDK=3')
110 CALL GTX(.1, ORY,'IPAT=DDDDDDDDDDDDDDDK=4')
120 CALL GTX(.1, ORY,'IPAT=DDDDDDADADDADDDDK=5')
CONTINUE
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
AS APOSTROPHE AND DOLLAR SIGN')
C ADVANCE FRAME
CALL FRAME
WRITE (6,1001)
1001 FORMAT (' DASHCHAR TEST SUCCESSFUL',24X,
'SEE PLOT TO VERIFY PERFORMANCE')
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END

April 1986

EXAMPIES
DEMONSTRATION PLOT FOR DASHCHAR

IPAT=DADADADADADADADK=1

IPAT=DDDDDDADADDADDDDK=2

IPAT=DDDDDADDDDDADDADK=3

IPAT=DDDDDDDDADDADDDDK=4

IPAT=DDDDDDDDDDADDADDDK=5

IPAT STRINGS. A AND D SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SI
PROGRAM TDASHL
C
C PURPOSE TO PROVIDE A DEMONSTRATION OF DASHLINE
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING FIVE CURVES EACH USING A DIFFERENT DASHCHAR PATTERNS IS PRODUCED.
C
C REQUIRES ROUTINES NONE
C
C X CONTAINS ABSCISSAE AND Y THE ORDINATE V ALUES OF THE CURVE DIMENSION X(31) ,Y(31) ,IPAT(5)
C
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C SELECT NORMALIZATION TRANSFORMATION 0 CALL GSEMNT(0)
C
C SET SOLID DASH PATTERN, 1111111111111111 (BINARY). BOOLEAN OPERATIONS (EMPLOYING LOCALLY IMPLEMENTED SUPPORT ROUTINES) ARE USED.
ISOLID = IOR (ISHIFT (32767,1), 1)
C
C ARRAY IPAT CONTAINS 5 DIFFERENT 16-BIT DASH PATTERNS. THE PATTERNS CONSTRUCTED WITH BOOLEAN OPERATIONS AS ABOVE.
The BINARY REPRESENTATIONS OF THE PATTERNS ARE
C
C 0001110001111111
C 1111000011110000
C 1111110011111100
C 1111111100000000
C 11111111111100
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 CENTRAL AXIS FOR EACH CURVE CALL FRSTD (.50,ORG-0.03)
CALL VECTD (.50,ORG+0.03)
CALL LINED (.109,ORG,.891,ORG)
CALL DASHDB (IPAT(K))
C COMPUTE VALUES FOR AND DRAW THE KTH CURVE 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 SET TEXT ALIGNMENT TO CENTER THE STRING AT THE LEFT OF THE STRING AND IN THE VERTICAL CENTER.
CALL GSTXAL(1,3)
C SET CHARACTER HEIGHT CALL GSCHH(.012)
C LABEL EACH CURVE WITH THE APPROPRIATE REPRESENTATION.
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=1111111000000000')
GO TO 70
60 CALL GTX (.1,ORY,'IPAT=1111111111111000')
70 CONTINUE
   CALL GSTXAL(2,3)
   CALL GTX (.5,.985,'DEMONSTRATION PLOT FOR DASHLINE')
C ADVANCE FRAME
   CALL FRAME
   WRITE (6,1001)
1001 FORMAT (' DASHLINE TEST SUCCESSFUL',24X,
               'SEE PLOT TO VERIFY PERFORMANCE')
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
   CALL GDAWK(1)
   CALL GCLWK(1)
   CALL GCLKS
STOP
END

~April 1986 -23- EXAMPLES
April 1986

~April 1986 -23- EXAMPLES
April 1986
PROGRAM TDASHS

C PURPOSE TO PROVIDE A DEMONSTRATION OF DASHSMTH

C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
CONTAINING FIVE CURVES EACH USING A DIFFERENT
DASHSMTH PATTERNS IS PRODUCED.

C REQUIRED ROUTINES NONE

C X CONTAINS ABCISSAE AND Y THE ORDINATE VALUES OF THE CURVE
DIMENSION X(31) ,Y(31)

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C SELECT NORMALIZATION TRANSFORMATION 0
CALL GSELTN(0)

C SET SOLID DASH PATTERN, 1111111111111111 (BINARY).
C BOOLEAN OPERATIONS (EMPLOYING LOCALLY IMPLEMENTED SUPPORT
C ROUTINES) ARE USED FOR PORTABILITY TO HOSTS WITH 16 BIT
C INTEGERS.

ISOLID = IOR (ISHIFT (32767,1), 1)
DO 130 K=1,5
   CALL DASHDB (ISOLID)
ORG =1.07-0.195*K

C DRAW CENTRAL AXIS FOR EACH CURVE
CALL FRSTD (.50,ORG-0.03)
CALL VECTD (.50,ORG+0.03)
CALL LASTD
CALL LINED (.109,ORG,.891,ORG)

C CALL SUBROUTINE DASHDC WITH A DIFFERENT DASHED LINE AND CHARACTER
C COMBINATION FOR EACH OF FIVE CURVES
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 COMPUTE VALUES FOR AND DRAW THE KTH CURVE
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 SET TEXT ALIGNMENT TO CENTER THE STRING AT THE LEFT OF THE
C STRING AND IN THE VERTICAL CENTER.
CALL GSTXAL(1,3)

C SET CHARACTER HEIGHT
CALL GSCHH(.012)

C LABEL EACH CURVE WITH THE APPROPRIATE CHARACTER REPRESENTATION.
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=DDDDDDADADDDDDDK=2')

April 1986

EXAMPLES
GO TO 130
100 CALL GTX(.1,ORY,'IPAT=DDDDADDDDDADDADK=3')
GO TO 130
110 CALL GTX(.1,ORY,'IPAT=DDDDAAAAADDADDDDK=4')
GO TO 130
120 CALL GTX(.1,ORY,'IPAT=DD_ADADDADDADDDDK=5')
130 CONTINUE
   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')
C
C ADVANCE FRAME
   CALL FRAME
   WRITE (6,1001)
1001 FORMAT (' DASHSMTH TEST SUCCESSFUL',24X,
   1 'SEE PLOT TO VERIFY PERFORMANCE')
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
   CALL GDAWK (1)
   CALL GCLWK(1)
   CALL GCLKS
   STOP
   END
DEMONSTRATION PLOT FOR DASHSMTH

IPAT=DADADADADADADAADK=1

IPAT=DDDDDDADADDDDDDDK=2

IPAT=DDDDADDDADDDDDDDK=3

IPAT=DDDDAAAAADDDDDDDK=4

IPAT=DDDDDDDDADDDDDDDK=5

IPAT STRINGS, A AND D SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SIGN

April 1986

-29-

EXAMPLES
PROGRAM TDASHP

C PURPOSE
TO PROVIDE A DEMONSTRATION OF DASHSUPR

C I/O
IF THE TEST IS SUCCESSFUL, A MESSAGE IS
PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
CONTAINING FIVE CURVES EACH USING A DIFFERENT
DASHSUPR PATTERNS IS PRODUCED.

C REQUIRED ROUTINES
NONE

C X CONTAINS ABSCISSAE AND Y THE ORDINATE VALUES OF THE CURVE
DIMENSION X(31), Y(31)

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C SELECT NORMALIZATION TRANSFORMATION 0
CALL GSELNT(0)

C RESET Initializes the model picture array and should be called with
C each new frame and before the other subroutines of the DASHSUPR
C package.
CALL RESET

C SET SOLID DASH PATTERN, 1111111111111111 (BINARY). BOOLEAN
C OPERATIONS (EMPLOYING LOCALLY IMPLEMENTED PLOT PACKAGE SUPPORT
C ROUTINES) ARE USED FOR PORTABILITY TO HOSTS WITH 16 BIT INTEGERS.
ISOLID = IOR (IShift (32767,1), 1)

C DO 130 K=1,5
    CALL DASHDB (ISOLID)
    ORG = 1.070.195*K

C DRAW CENTRAL AXIS FOR EACH CURVE
CALL FRSTD (.50, ORG-0.03)
CALL VECTD (.50, ORG+0.03)
CALL LASTD
CALL LINED (.109, ORG,.891, ORG)

C CALL SUBROUTINE DASHDC WITH A DIFFERENT PATTERN FOR EACH CURVE.
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 COMPUTE VALUES FOR AND DRAW THE KTH CURVE
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 CALL CURVED (X,Y,31)

C SET TEXT ALIGNMENT TO CENTER THE STRING AT THE LEFT OF THE
C STRING AND IN THE VERTICAL CENTER
CALL GSTXAL(1,3)

C SET CHARACTER HEIGHT
CALL GSCHH(.012)

C LABEL EACH CURVE WITH THE APPROPRIATE CHARACTER REPRESENTATION
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=DDDDDDADADDDDDDK=2')
GO TO 130
100 CALL GTX(.1, ORY, 'IPAT=DDDDADDDDDADDDAK=3')
GO TO 130
110 CALL GTX(.1, ORY, 'IPAT=DDDDAADAADDDDDDK=4')
GO TO 130
120 CALL GTX(.1, ORY, 'IPAT=DDDADDDADDDDDADK=5')
130 CONTINUE

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')

C
C ADVANCE FRAME
CALL FRAME
WRITE (6, 1001)
1001 FORMAT (' DASHSUPR TEST SUCCESSFUL', 24X,
1 'SEE PLOT TO VERIFY PERFORMANCE')

C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END
DEMONSTRATION PLOT FOR DASHSUPR

IPAT=DADADADADADADADK=1

IPAT=DDDDADADADDDDDDK=2

IPAT=DDDDADDDDDADDDAK=3

IPAT=DDDDAAAAADDDDDDK=4

IPAT=DDDAADDDDDADDDAK=5

IPAT STRINGS, A AND D SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SI
PROGRAM TEZMAP
C
C PURPOSE      PROVIDE A SIMPLE DEMONSTRATION OF EZMAP
C
C I/O          IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C              PRINTED ON UNIT 6 TO THAT EFFECT AND
C              TEN CONTINENTAL OUTLINE PLOTS, EACH
C              RESULTING FROM A DIFFERENT SPECIFIED
C              PROJECTION, ARE PRODUCED.
C
C REQUIRED ROUTINES   DASHLINE
C
C REQUIRED FILES   UNIT 1 MUST BE OPENED. IT CONTAINS THE
C                  EZMAP DATASET.

DIMENSION PLM1(2),PLM2(2),PLM3(2),PLM4(2)
COMMON /ERROR/ IFRAME, IERRR
DATA PLM1,PLM2,PLM3,PLM4 /8*0.0/

C SPECIFY COORDINATES FOR PLOT TITLES. ON AN ABSTRACT PLOTTER GRID
C WHERE THE COORDINATES RANGE FROM 0.0 TO 1.0, THE VALUES TX
C AND TY DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/0.5/, TY/0.9765/

C TURN ON ERROR RECOVERY MODE
CALL ENTSR(IDUM,1)

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C OPEN FILE WHICH CONTAINS MAP DATA. THIS CALL IS INSTALLATION
C DEPENDENT.

OPEN (UNIT=1,FILE='ezmap.out',STATUS='OLD',FORM='UNFORMATTED')

C FRAME 1
IFRAME = 1

C SET THE PROJECTION-TYPE PARAMETER
CALL MAPROJ('ST',80.0,-160.0,0.0)

C SET THE LIMIT PARAMETERS
CALL MAPSET('MA',PLM1,PLM2,PLM3,PLM4)

C SET THE OUTLINE-DATASET PARAMETER
CALL MAPSTC('OU','PS')

C DRAW THE MAP
CALL MAPDRW
IF( NERRO(IERR).NE.0) CALL RPTERR

C SELECT NORMALIZATION TRANS TO WRITE TITLE
CALL GSELNT(0)
CALL WTISTR(TX,TY,'EZMAP DEMONSTRATION: STEREOGRAPHIC PROJECTION'
+       ',2,0,0)
CALL FRAME

C FRAME 2
IFRAME = 2

C SET THE PROJECTION-TYPE PARAMETER
CALL MAPROJ('OR',60.0,-120.0,0.0)

April 1986 -34- EXAMPLES
C DRAW THE MAP
   CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
C
C WRITE THE TITLE
   CALL GSELTNT(0)
   CALL WTSTR(TX,TY,
   +   'EZMAP DEMONSTRATION: ORTHOGRAPHIC PROJECTION',2,0,0)
   CALL FRAME
20 CONTINUE
C
C FRAME 3
   IFRAME = 3
C
C 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')
C
C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
   CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELTNT(0)
   CALL WTSTR(TX,TY,
   +   'EZMAP DEMONSTRATION: LAMBERT CONFORMAL CONIC PROJECTION',
   +   2,0,0)
   CALL FRAME
30 CONTINUE
C
C FRAME 4
   IFRAME = 4
C
C SET THE PROJECTION-TYPE, LIMITS, AND OUTLINE-DATASET PARAMETERS
   CALL MAPROJ('LE',20.0,-40.0,0.0)
   CALL MAPSET('MA',PLM1,PLM2,PLM3,PLM4)
   CALL MAPSTC('OU','CO')
C
C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
   CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELTNT(0)
   CALL WTSTR(TX,TY,
   +   'EZMAP DEMONSTRATION: LAMBERT EQUAL AREA PROJECTION',2,0,0)
   CALL FRAME
40 CONTINUE
C
C FRAME 5
   IFRAME = 5
C
C SET THE MAP PROJECTION-TYPE PARAMETER
   CALL MAPROJ('GN',0.0,0.0,0.0,0.0)
C
C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
   CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELTNT(0)
   CALL WTSTR(TX,TY,
   +   'EZMAP DEMONSTRATION: GNOMONIC PROJECTION',2,0,0)
   CALL FRAME
50 CONTINUE
C
C FRAME 6
   IFRAME = 6
C
April 1986 -35- EXAMPLES
C SET THE MAP PROJECTION-TYPE PARAMETER
CALL MAPROJ('AE',-20.0,40.0,0.0)

C
C SET THE GRID SPACING
CALL MAPSTR('GR',5.0)

C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELENT(0)
   CALL WTSTR(TX,TY,
      + 'EZMAP DEMONSTRATION: AZIMUTHAL EQUIDISTANT PROJECTION',
      + 2,0,0)
   CALL FRAME
60 CONTINUE

C FRAME 7
IFRAME = 7

C SET THE MAP PROJECTION TYPE PARAMETER
CALL MAPROJ('CE',-40.0,80.0,0.0)

C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELENT(0)
   CALL WTSTR(TX,TY,
      + 'EZMAP DEMONSTRATION: CYLINDRICAL EQUIDISTANT PROJECTION',
      + 2,0,0)
   CALL FRAME
70 CONTINUE

C FRAME 8
IFRAME = 8

C SET THE MAP PROJECTION-TYPE PARAMETER
CALL MAPROJ('ME',-60.0,120.0,0.0)

C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELENT(0)
   CALL WTSTR(TX,TY,'EZMAP DEMONSTRATION: MERCATOR PROJECTION',
      + 2,0,0)
   CALL FRAME
80 CONTINUE

C FRAME 9
IFRAME = 9

C SET THE MAP PROJECTION-TYPE PARAMETER
CALL MAPROJ('MO',-80.0,160.0,0.0)

C DRAW THE ABOVE DEFINED MAP AND WRITE THE TITLE
CALL MAPDRW
   IF( NERRO(IERR).NE.0) CALL RPTERR
   CALL GSELENT(0)
   CALL WTSTR(TX,TY,'EZMAP DEMONSTRATION: MOLLWEIDE-TYPE
      PROJECTION',2,0,0)
   CALL FRAME
90 CONTINUE

C FRAME 10
IFRAME = 10

C SET THE MAP PROJECTION-TYPE PARAMETER
EXAMPLES
April 1986 -36-
CALL MAPROJ('SV',0.0,-135.0,0.0)

C
C SET SATELLITE DISTANCE AND SUPPRESS GRID LINES
CALL MAPSTR('SA',6.631)
CALL MAPSTI('GR',0)

C
C DRAW THE ABOVE DEFINED MAP AND WRITE TITLE
CALL MAPDRW
IF(NERRO(IERR).NE.0) CALL RPTERR
CALL GSELNT(0)
CALL WTSTR(TX,TY,
  'EZMAP DEMONSTATION: SATELLITE VIEW PROJECTION',2,0,0)
CALL FRAME
100 CONTINUE

C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS

C
IF(IERRR.EQ.0) WRITE(6,1000)
IF(IERRR.EQ.1) WRITE(6,1001)
IERROR = IERR
STOP

C
1000 FORMAT(' EZMAP TEST SUCCESSFUL',24X,
  'SEE PLOT TO VERIFY PERFORMANCE')
1001 FORMAT(' EZMAP TEST UNSUCCESSFUL')
END

SUBROUTINE RPTERR
C
C ROUTINE TO REPORT ERROR MESSAGES
C
COMMON /ERROR/ IFRAME,IERRR
WRITE(6,1000) IFRAME
CALL EPRIN
WRITE(6,1001)
CALL ERROF
IERRR = 1
RETURN
1000 FORMAT(' ERROR IN FRAME ',I2,5X,'ERROR MESSAGE FOLLOWS:')
1001 FORMAT(' ******,//')
END
EZMAP DEMONSTRATION: LAMBERT EQUAL AREA PROJECTION
EZMAP DEMONSTRATION, GNOMONIC PROJECTION
EZMAP DEMONSTRATION: AZIMUTHAL EQUIDISTANT PROJECTION
EZMAP DEMONSTRATION: CYLINDRICAL EQUIDISTANT PROJECTION

April 1986
EZMAP DEMONSTRATION, MOLLWEIDE-TYPE PROJECTION
EZMAP DEMONSTATION: SATELLITE VIEW PROJECTION
PROGRAM TGRIDA

C PURPOSE
PROVIDES A SIMPLE DEMONSTRATION OF GRIDAL

C I/O
IF THE TEST IS SUCCESSFUL, A MESSAGE IS
PRINTED ON UNIT 6 TO THAT EFFECT AND 4
FRAMES ARE PRODUCED.

C REQUIRED ROUTINES
NONE

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C SET NORMALIZATION TRANSFORMATION TO 1
CALL GSWN(1.0,1.0,1.0,1.0)
CALL GSVP(1.2,1.8,2.8)

C GRID
CALL GSELLNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR GRID',2,0,0)
CALL GSELLNT(1)
CALL GRID(5,2,6,3)
CALL FRAME

C TICK4
CALL GSELLNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR TICK4',2,0,0)
CALL GSELLNT(1)
CALL TICK4(150,50,150,50)
CALL PERIML(5,2,6,3)
CALL FRAME
CALL TICK4(12,8,12,8)

C LABMOD
CALL GSELLNT(0)
CALL WTSTR(.5,.9,'DEMONSTRATION PLOT FOR LABMOD',2,0,0)
CALL GSELLNT(1)
CALL LABMOD('(E10.2)', '(F4.2)', 10, 4, 15, 15, 0, 0, 0)
CALL HALFAX(2,1,10,1,0,0,0,1,1)
CALL FRAME

C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS
CALL GDWK(1)
CALL GCLWK(1)
CALL GCLKS

1001 FORMAT(I2)
WRITE(6,600)
600 FORMAT('GRIDAL SUCCESSFUL--SEE PLOTS TO VERIFY PERFORMANCE')
STOP
END
DEMONSTRATION PLOT FOR GRID
DEMONSTRATION PLOT FOR TICK4

April 1986
DEMONSTRATION PLOT FOR LABMOD

0.500E+00 0.10E+01
April 1986

EXAMPLES

1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.
0.

-51-
PROGRAM THAFTO
C
C PURPOSE  PROVIDES A SIMPLE DEMONSTRATION OF HAFTON
C
C I/O  IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND A
C HALF-TONE PLOT IS PRODUCED.
C
C REQUIRED ROUTINES  NONE
C
C Z IS THE ARRAY TO BE PLOTTED.
REAL Z(21,25)
C
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM
C 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/0.0762/, TY/0.9769/
C
C SPECIFY THE LOW (FLO) AND HIGH (FHI) CONTOUR VALUES FOR HAFTON;
C ALSO THE NUMBER OF UNIQUE CONTOUR LEVELS. NOPT DETERMINES
C HOW Z IS MAPPED ONTO THE INTENSITIES AND THE DIRECTNESS OF MAPPING.
DATA FLO/-4.0/, FHI/4.0/, NLEV/8/, NOPT/-3/
C
C FILL THE ARRAY TO BE PLOTTED
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 OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C SELECT NORMALIZATION TRANS 0 FOR PLOTTING TITLE
CALL GSELNT (0)
C
C ENTRY EZHFTN REQUIRES ONLY THE ARRAY NAME AND ITS DIMENSIONS
C FOR EXAMPLE:
C   CALL EZHFTN (Z,21,25)
C
C ENTRY HAFTON ALLOWS USER SPECIFICATIONS OF PLOT PARAMETERS.
C THE PLOT IS TITLED BEFORE CALLING HAFTON.
CALL WTSTR (TX,TY,
           'DEMONSTRATION PLOT FOR ENTRY HAFTON OF HAFTON',2,0,-1)
CALL HAFTON (Z,21,21,25,FLO,FHI,NLEV,NOPT,0,0,0.)
CALL FRAME
C
C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS
CALL GDWK(1)
CALL GCLWK(1)
CALL GCLKS
WRITE (6,1001)
1001 FORMAT(' HAFTON SUCCESSFUL',24X,'SEE PLOT TO CHECK PERFORMANCE')
STOP
END

April 1986

-52-

EXAMPLES
DEMONSTRATION PLOT FOR ENTRY HAFTON OF HAFTON

April 1986

-53-

EXAMPLES
PROGRAM THSTGR

C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF HSTGRM
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE TO THAT
C EFFECT IS WRITTEN TO UNIT 6 AND 1 HISTOGRAM
C FRAME IS PRODUCED.
C
C REQUIRED ROUTINES NONE
C
C ARRAY ARR IS FILLED WITH VALUES TO BE USED AS INPUT FOR HSTGRM
REAL INARR(100), X, ARR7(7), WORK(212)
INTEGER IWORK(84)
DATA TX / .5/, TY / .9765/

C DO 10 I = 1, 100
   X = FLOAT(I)
   INARR(I) = 10. * ALOG10(X)
10 CONTINUE

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1, 2, 1)
CALL GACWK(1)

C NPTS = 100
NCLASS = 11
RLO = 2.0
RHI = 19.7
LWORK = 8 * (NCLASS + 3) + NPTS
LIWORK = 6 * (NCLASS + 3)

C DEMONSTRATION OF THE DEFAULT VERSION OF HSTGRM, NOT ACTUALLY CALLED
CALL HSTGRM(INARR, NPTS, RLO, RHI, NCLASS, WORK, LWORK,
+ IWORK, LIWORK)

C DEMONSTRATE HSTGRM WITH WINDOWING OPTION ON AND SHADING AND
C FRAME OPTIONS OFF
C (CALL HSTOPL('DEF=ON') TO RESET ALL OPTIONS TO DEFAULT)
CALL HSTOPL('DEF=ON')
CALL HSTOPC('TI=ON', -OPTIONS CHANGED: LAB, FQN, TIT, PER, FOR AND WIN')
ARR7(1) = .2
ARR7(2) = .8
ARR7(3) = 0.
ARR7(4) = .5
CALL HSTOPR('WIN=ON', ARR7, 4)
CALL HSTOPL('PER=OFF')
CALL HSTOPC('LAB=ON', 'A DIFFERENT USER LABEL')
CALL HSTOPC('FQN=ON', 'NUMBER OF OCCURRENCES IN EACH CLASS')
CALL HSTOPC('FOR=ON', '(F10.5)')
NCLASS = 11
print *, 'i am here'
CALL HSTGRM(INARR, NPTS, RLO, RHI, NCLASS, WORK, LWORK,
+ IWORK, LIWORK)
WRITE (6, 1001)
1001 FORMAT ('HSTGRM TEST SUCCESSFUL', 24X,
+ 'SEE PLOT TO VERIFY PERFORMANCE')

C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END

April 1986

-54-
PROGRAM TISOSR
C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF ISOSRF
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE SAMPLE PLOT IS PRODUCED.
C REQUIRED ROUTINES NONE
C
DIMENSION T(21,31,19),SLAB(33,33),EYE(3)
DATA EYE /100.,150.,125./
DATA NU,NV,NW,IX,IY,TISO,MUVWP2,IFLAG/21,31,19,44,95,0.,33,-7/
DATA RBIG1,RBIG2,RSML1,RSML2/6.,6.,2.,2./
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING. FILL THE ARRAY
DATA CX,CY /0.50,.95/
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+FKMID2*FKMID2+.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,FIMID*FIMID+FIP2*FJP2+FKP2-FKP2-RSML2*RSML2)
   10 CONTINUE
20 CONTINUE
30 CONTINUE
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C AN EXAMPLE OF THE EZISOS CALL IS PRESENTED BELOW BUT IS NOT CALLED:
C CALL EZISOS (T,NU,NV,NW,EYE,SLAB,TISO)
C TEST ISOSRF WITH SUBARRAY OF T
CALL WSTR(CX,CY,'DEMONSTRATION PLOT FOR ENTRY ISOSRF',2,0,0)
MU=NU/2
MV=NV/2
MW=NW/2
MUVWP2=MAX0(MU,MV,MW)+2
CALL ISOSRF(T(MU,MV,MW),NU,MV,MW,EYE,MUVWP2,SLAB,TISO,IFLAG)
CALL FRAME
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
WRITE (6,1001)
STOP
1001 FORMAT(' ISOSRF SUCCESSFUL',24X,'SEE PLOT TO CHECK PERFORMANCE')
END
DEMONSTRATION PLOT FOR ENTRY ISOSRF
PROGRAM TISOHR

C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF ISOSRFHR
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE SAMPLE PLOT IS PRODUCED.
C REQUIRED ROUTINES NONE
C REQUIRED FILES UNIT 7 MUST BE ASSIGNED AS A SCRATCH UNIT.

DIMENSION EYE(3), S(4), IS2(4,200), ST1(81,51,2), IOBJS(81,51)
COMMON /UNITS/ IUNIT

C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING. FILL THE ARRAY
DATA CX, CY / .50, .95/, IUNIT/9/
DATA EYE(1), EYE(2), EYE(3) / 200., 250., 250. /

DATA NU, NV, NW / 51, 81, 51 /
DATA LX, NX, NY / 4, 180, 180 /
DATA S(1), S(2), S(3), S(4) / 10., 1010., 10., 1010. /
DATA MV / 81 /

C SPECIFY THE LARGE AND SMALL RADII FOR THE INDIVIDUAL DOUGHNUTS
DATA RBIG1, RBIG2, RSML1, RSML2 / 20., 20., 6., 6. /

OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C OPEN SCRATCH FILE : /tmp/scrtch THIS OPEN IS AN INSTALLATION DEPENDENT CALL.
UNIT = 7
OPEN(UNIT=7,FILE='\tmp\scrtch',STATUS='SCRATCH', 1 FORM='UNFORMATTED')
CALL INIT3D (EYE,NU,NV,NW,ST1,LX,NY,IS2,IUNIT,S)

C CREATE AND PLOT DATA FOR TWO INTERLOCKING DOUGHNUTS
JCENT1 = FLOAT(NV)*.5-RBIG1*.5
JCENT2 = FLOAT(NV)*.5+RBIG2*.5
DO 70 IBKWD=1,NU
   I = NU+1-IBKWD
   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/(FJMID*FJMID+FJMID2*FJMID2+.1))
      FIP1 = (1.-F1)*FJMID
      FIP2 = (1.-F2)*FJMID
      FJP1 = (1.-F1)*FJMID1
      FJP2 = (1.-F2)*FJMID2
      FKP1 = (1.-F1)*FKMID
      FKP2 = (1.-F2)*FKMID
      TEMP = AMIN1(FJMID**2+FJP1**2+FKP1**2-RSML1**2,
                   FKMID**2+FIP2**2+FJP2**2-RSML2**2)
      IF (TEMP .LE. 0.) IOBJS(J,K) = 1

April 1986

EXAMPLES
IF (TEMP .GT. 0.) IOBJS(J,K) = 0
10 CONTINUE
20 CONTINUE

C
C SET PROPER WORDS TO 1 FOR DRAWING AXES
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
C CALL THE DRAW AND REMEMBER ROUTINE FOR THIS SLAB
   CALL DANDR (NV,NW,ST1,LX,NX,NY,IS2,IUNIT,S,IOBJS,MV)
70 CONTINUE

C
C TITLE THE PLOT
   CALL GQCNTN(IER,ICN)
   CALL GSELNT(0)
   CALL WTSTR(CX,CY,'DEMONSTRATION PLOT FOR ISOSRFHR',2,0,0)
   CALL GSELNT(ICN)

C
C ADVANCE THE PLOTTING DEVICE
   CALL FRAME

C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
   CALL GDAWK(1)
   CALL GCLWK(1)
   CALL GCLKS
   WRITE (6,1001)
   1001 FORMAT (' ISOSRFHR TEST SUCCESSFUL',24X,'SEE PLOT TO VERIFY PERFORMANCE')
STOP
END
DEMONSTRATION PLOT FOR ISOSRFHR
PROGRAM TPWRTX

C PURPOSE PROVIDES A DEMONSTRATION OF PWRITX

C I/O If the test is successful, a message is
C printed on unit 6 to that effect and 4 frames
C containing the sample plots are produced.
C The first three frames contain complete
C character plots, and the fourth frame
C tests various settings of the function
C codes.

C REQUIRED ROUTINES NONE

C REQUIRED FILES UNIT 3 MUST BE OPENED. THIS DATASET CONTAINS
C THE INPUT FOR THE PWRITX CHARACTER SETS.

C ALGORITHM 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
C column is labelled with its function code.
C The fifth plot invokes PWRITX with various
C function codes.

C DAT contains the standard character set
C
CHARACTER*1 DAT(48)
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', 'Q', 'I',
1 '2', '3', '4', '5', '6', '7', '8',
2 '9', '+', '-', 'x', 'y', 'z', '1',
3 '&', '=' ' ', ' ', ' ', ' ', ' ', '/

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C Use normalization transformation 0
CALL GSELNT(0)

C OPEN BINARY DATA FILE ON UNIT 3 TO BE USED WITH PWRITX. THIS CALL
C IS INSTALLATION DEPENDENT.
OPEN(UNIT=3,FILE='pcrbin.out',STATUS='OLD',FORM='UNFORMATTED')

C A separate frame is produced for each iteration through this loop
DO 160 K=1,4
C Label the column and change the function code
DO 150 J=1,12

April 1986

-62-
\texttt{XPOS = FLOAT(J*80-39) / 1024.}
\texttt{GO TO ( 10, 20, 30, 40, 50, 60, 70, 80, 90, 100,}
\texttt{110,120),J}
\texttt{10 CALL PWRITX (XPOS,.9375,'PRU',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''PRU''',5,1,0,0)
GO TO 130}
\texttt{20 CALL PWRITX (XPOS,.9375,'PRL',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''PRL''',5,1,0,0)
GO TO 130}
\texttt{30 CALL PWRITX (XPOS,.9375,'IRU',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''IRU''',5,1,0,0)
GO TO 130}
\texttt{40 CALL PWRITX (XPOS,.9375,'IRL',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''IRL''',5,1,0,0)
GO TO 130}
\texttt{50 CALL PWRITX (XPOS,.9375,'KRU',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''KRU''',5,1,0,0)
GO TO 130}
\texttt{60 CALL PWRITX (XPOS,.9375,'KRL',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''KRL''',5,1,0,0)
GO TO 130}
\texttt{70 CALL PWRITX (XPOS,.9375,'PGU',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''PGU''',5,1,0,0)
GO TO 130}
\texttt{80 CALL PWRITX (XPOS,.9375,'PGL',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''PGL''',5,1,0,0)
GO TO 130}
\texttt{90 CALL PWRITX (XPOS,.9375,'IGU',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''IGU''',5,1,0,0)
GO TO 130}
\texttt{100 CALL PWRITX (XPOS,.9375,'IGL',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''IGL''',5,1,0,0)
GO TO 130}
\texttt{110 CALL PWRITX (XPOS,.9375,'KGU',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''KGU''',5,1,0,0)
GO TO 130}
\texttt{120 CALL PWRITX (XPOS,.9375,'KGL',3,16,0,0,)
CALL PWRITX (1./1024.,1./1024.,''KGL''',5,1,0,0)
130 CONTINUE}
\texttt{C
\texttt{Draw twelve characters with the same function code}
\texttt{DO 140 I=1,12}
\texttt{\texttt{YPOS = FLOAT( 980-I*80 ) / 1024.}}
\texttt{\texttt{IF = I+(K-1)*12.}}
\texttt{\texttt{CALL PWRITX (XPOS,YPOS,DAT(IF),1,1,0,0)}}
\texttt{140 CONTINUE}
\texttt{C
\texttt{Return function to Principle Roman Upper to label column}}
\texttt{\texttt{CALL PWRITX (1./1024.,1./1024.,''PRU''',5,1,0,0)}}
\texttt{\texttt{150 CONTINUE}}
\texttt{C
\texttt{Label frame}}
\texttt{\texttt{CALL PWRITX(.5,1000./1024.,}}
\texttt{\texttt{''DEMONSTRATION PLOT FOR PWRITX'',29,1,0,0)}}
\texttt{\texttt{\texttt{CALL FRAME \texttt{160 CONTINUE}}}}
\texttt{C
\texttt{Test more function codes.}}
\texttt{\texttt{Tests:}}
\texttt{\texttt{Upper and lower case function codes}}
\texttt{\texttt{Sub- and Super-scripting and Normal function codes}}
\texttt{\texttt{Down and Across function codes}}
\texttt{\texttt{Direct Character access}}
\texttt{\texttt{X and Y coordinate control, Carriage control function codes}}
\texttt{\texttt{Orientation of string (argument to PWRITX)}}

April 1986

-63-

EXAMPLES
C 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)
C
C Upper and Lower case
CALL PWRITX(.65,.25,'''L2''LOWER, 3 '''U3''UPPER',24,1,0,0)
C
C Level definitions (sub and superscripting)
CALL PWRITX(.95,.15,'''U1''S''S''UPERSCRIPTING',29,0,0,1)
CALL PWRITX(.95,.1,'''L1''S''B''UBSCRIPTING',
$ 30,0,0,1)
CALL PWRITX(.95,.05,'''N''SHOW '''U1''U''S''SE OF '''NU''NORMAL',
$ 31,0,0,1)
C
C Direction definitions
CALL PWRITX(.05,.5,'''D''WNA''A''CROSS',16,0,0,-1)
C
C 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''.LEFT',
$ 37,14,90,-1)
CALL PWRITX(.8,8,'''L3''USE C''CL''FOR''C''CARRIAGE''C''RETURNS',
$ 37,16,0,0)
CALL PWRITX(.1,1,'''UX50Y50''( X50, Y50 )''X99Y99''( X99, Y99 )',
$ 41,14,0,-1)
C
CALL FRAME
WRITE (6,1001)
1001 FORMAT ( 'PWRITX TEST SUCCESSFUL',24X,
$ 'SEE PLOT TO VERIFY PERFORMANCE')
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS.-
STOP'
END
EXAMPLES
April 1986
-64-
DEMONSTRATION PLOT FOR PWRITX

<table>
<thead>
<tr>
<th>PRU</th>
<th>PRL</th>
<th>IRU</th>
<th>IRL</th>
<th>KRU</th>
<th>KRL</th>
<th>PGU</th>
<th>PGL</th>
<th>IGU</th>
<th>IGL</th>
<th>KGU</th>
<th>KGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>m</td>
<td>M</td>
<td>m</td>
<td>M</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>n</td>
<td>N</td>
<td>n</td>
<td>N</td>
<td>Ξ</td>
<td>ξ</td>
<td>ξ</td>
<td>ξ</td>
<td>ξ</td>
<td>ξ</td>
<td>ξ</td>
</tr>
<tr>
<td>O</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>/</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Q</td>
<td>q</td>
<td>q</td>
<td>q</td>
<td>q</td>
<td>q</td>
<td>P</td>
<td>ρ</td>
<td>ρ</td>
<td>ρ</td>
<td>ρ</td>
<td>ρ</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>Σ</td>
<td>σ</td>
<td>σ</td>
<td>σ</td>
<td>σ</td>
<td>σ</td>
<td>σ</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>T</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
<td>τ</td>
</tr>
<tr>
<td>U</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>Φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>X</td>
<td>χ</td>
<td>χ</td>
<td>χ</td>
<td>χ</td>
<td>χ</td>
<td>χ</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>O</td>
<td>Ψ</td>
<td>ψ</td>
<td>ψ</td>
<td>ψ</td>
<td>ψ</td>
<td>ψ</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>□</td>
<td>Ω</td>
<td>Ω</td>
<td>Ω</td>
<td>Ω</td>
<td>Ω</td>
<td>Ω</td>
</tr>
</tbody>
</table>

April 1986
EXAMPLES
### DEMONSTRATION PLOT FOR PWRTTX

<table>
<thead>
<tr>
<th>PRU</th>
<th>PRL</th>
<th>IRU</th>
<th>IRL</th>
<th>KRU</th>
<th>KRL</th>
<th>PGU</th>
<th>PGL</th>
<th>IGU</th>
<th>IGL</th>
<th>KGU</th>
<th>KGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y Y y y ∆</td>
<td>O O O O O O O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z z z z z</td>
<td>✤ O O S ≈ Π 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 : 0 : 0 : 0 → 0 → 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 ; 1 ; 1 ; 1 ↑ 1 ↑ 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 ! 2 ! 2 ! 2 ← 2 ← 2 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 ? 3 ? 3 ? 3 ↓ 3 ↓ 3 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 × 4 × 4 × 4 ' 4 ' 4 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 ÷ 5 ÷ 5 ÷ 5 ' 5 ' 5 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 ≡ 6 ≡ 6 ≡ 6 ' 6 ' 6 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 ≡ 7 ≡ 7 ≡ 7 ' 7 ' 7 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 &lt; 8 &lt; 8 &lt; 8 &lt; 8 8 8 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 &gt; 9 &gt; 9 &gt; 9 &lt; 9 &lt; 9 9 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

April 1986

-67-

**EXAMPLES**
<table>
<thead>
<tr>
<th>PRU</th>
<th>PRL</th>
<th>IRU</th>
<th>IRL</th>
<th>KRU</th>
<th>KRL</th>
<th>PGU</th>
<th>PGL</th>
<th>IGU</th>
<th>IGL</th>
<th>KGU</th>
<th>KGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>≤</td>
<td>+</td>
<td>≤</td>
<td>+</td>
<td>.</td>
<td>&amp;</td>
<td>#</td>
<td>&amp;</td>
<td>#</td>
<td>.</td>
<td>#</td>
</tr>
<tr>
<td>−</td>
<td>≥</td>
<td>−</td>
<td>≥</td>
<td>−</td>
<td>*</td>
<td>C</td>
<td>≥</td>
<td>C</td>
<td>≥</td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>∂</td>
<td>*</td>
<td>∂</td>
<td>*</td>
<td>_</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>_</td>
<td>/</td>
<td>_</td>
<td>/</td>
<td>_</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>√</td>
<td>$</td>
<td>√</td>
<td>$</td>
<td>√</td>
<td>$</td>
<td>√</td>
<td>$</td>
<td>√</td>
<td>$</td>
<td>√</td>
</tr>
<tr>
<td>=</td>
<td>f</td>
<td>=</td>
<td>f</td>
<td>=</td>
<td>f</td>
<td>∈</td>
<td>f</td>
<td>∈</td>
<td>f</td>
<td>→</td>
<td>f</td>
</tr>
</tbody>
</table>

April 1986

-68-
use V for vertical steps

use C for carriage returns

DO W N ACROSS

2 lower, 3 upper

(this is superscripting
THIS IS SUBSCRIPTING
SHOW USE OF NORMAL

( X99, Y99 )
( X50, Y50 )
PROGRAM TPWRY

C PURPOSE
PROVIDES A SIMPLE DEMONSTRATION OF PWRITY

C I/O
IF THE TEST IS SUCCESSFUL, A MESSAGE IS
PRINTED ON UNIT 6 TO THAT EFFECT AND A FRAME
CONTAINING THE SAMPLE PLOT IS PRODUCED.

C REQUIRED Routines
NONE

C

OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C DEFINE NORMALIZATION TRANS 1 AND LOG SCALING
CALL SET(0.0, 1.0, 0.0, 1.0, 0.0, 1024.0, 1.0, 1024.0, 1)

C LABEL FRAME
CALL PWRITY(512.0,950.0,
1 'DEMONSTRATION PLOT FOR PWRITY', 29, 2, 0, 0)

C 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)
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 TEST PWRITY FOR DIFFERENT CHARACTER ORIENTATIONS.
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 TEST CENTERING OPTIONS FOR PWRITY.
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)
1001 FORMAT ('PWRITY TEST SUCCESSFUL',24X,
1 'SEE PLOT TO VERIFY PERFORMANCE')

C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END

April 1986
DEMONSTRATION PLOT FOR PWRITY

SIZE TEST
SIZE TEST
SIZE TEST
SIZE TEST
SIZE TEST

CENTR TEST
CENTR TEST
CENTR TEST
PROGRAM TPWRZI
C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF PWRZI
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE CONTOUR PLOT IS PRODUCED.
C
C ISOSRF IS CALLED; PWRZI IS CALLED 3 TIMES TO
C LABEL THE FRONT, SIDE, AND BACK OF THE PICTURE
C
C REQUIRED ROUTINES ISOSRF
C
C DIMENSION T(21,31,19),SLAB(33,33),EYE(3)
C SPECIFY COORDINATES FOR PLOT TITLES. THEN SET OTHER VARIABLES.
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, MUVWP2, IFLAG, EYE/0., 33., -7., 100., 150., 125./

C FILL THREE DIMENSIONAL ARRAY TO BE PLOTTED
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,
                        FJMID*FJMID+FIP2*FIP2+FJP2*FJP2-RSML2*RSML2)
     10 CONTINUE
   20 CONTINUE
30 CONTINUE

C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

C TEST ISOSRF WITH SUBARRAY OF T
CALL WTSTR (TX,TY,'DEMONSTRATION PLOT FOR PWRZI',2,0,0)
MU = NU/2
MV = NV/2
MW = NW/2
MUVWP2 = MAX0(MU,MV,MW)+2
CALL ISOSRF (T(MU,MV,MW),NU,MU,NV,MV,MW,EYE,MUVWP2,SLAB,TISO,
   1 IFLAG)
CALL PWRZI (5.,16.,.5,'FRONT',5,35,-1,3,0)
CALL PWRZI (11.,7.5,.5,'SIDE',4,35,2,-1,0)
CALL PWRZI (5.,1.,5.,' BACK BACK BACK BACK BACK BACK',25,35,-1,3,0)
CALL FRAME

C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
WRITE (6,1001)
1001 FORMAT (' PWRZI SUCCESSFUL',24X,'SEE PLOT TO CHECK PERFORMANCE')
STOP
END

April 1986 -72-
DEMONSTRATION PLOT FOR PWRZI

BACK BACK BACK BACK BACK

SIDE  FRONT

April 1986

-73-

EXAMPLES
PROGRAM TPWRZS
C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF PWRZS
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE CONTOUR PLOT IS PRODUCED.
C AFTER SRFACE IS CALLED, PWRZ IS CALLED TO
C LABEL THE FRONT, SIDE, AND BACK OF THE PICTURE.
C
C REQUIRED ROUTINES SRFACE
C
C LOAD THE SRFACE COMMON BLOCK, NEEDED TO SURPRESS NEWFM CALL
C COMMON /SRFIP1/ IFR,ISTP,IROTS,IDRX,IDRY,
1 IDRZ,IUPPER,ISKIRT,NCLA,THETA,
2 HSKIRT,CHI,CLO,CINC,ISPVAL
C
C SPECIFY COORDINATES FOR PLOT TITLES, GRID LOOP INDICES, LINE OF SIGHT
DATA TX/0.4375/, TY/0.9667/
DATA M/20/, N/30/, S/4.,5.,3.,0.,0.,0./
C
C DEFINE FUNCTION VALUES AND STORE IN Z
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 SRFACE PARAMETERS TO SURPRESS FRAME CALL AND DRAW CONTOURS
IFR = 0
IDRZ = 1
C
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C SELECT NORMALIZATION TRANS NUMBER 0 AND LABEL THE PLOT
CALL GSELNT (0)
CALL WTSTR (TX,TY,'DEMONSTRATION PLOT FOR PWRZS',2,0,0)
C
C DRAW SURFACE PLOT AND PUT PWRZS LABELS ON THE PICTURE
CALL SRFACE (X,Y,Z,MM,M,M,N,S,0.)
CALL PWRZS (0.,1.1,0.,'FRONT',5,35,-1,3,0)
CALL PWRZS (1.1,0.,0.,'SIDE',4,35,2,-1,0)
CALL PWRZS (0.,-1.1,.2,'BACK BACK BACK BACK BACK',20,35,-1,3,0)
CALL FRAME
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
WRITE (6,1001)
1001 FORMAT(' PWRZS SUCCESSFUL',24X,'SEE PLOT TO CHECK PERFORMANCE')
STOP
END
April 1986

April 1986

April 1986

April 1986
DEMONSTRATION PLOT FOR PWRZS

BACK BACK BACK BACK

SIDE

FRONT
PROGRAM TPWRZT
C PURPOSE
PROVIDES A SIMPLE DEMONSTRATION OF PWRZT
C
C I/O
IF THE TEST IS SUCCESSFUL, A MESSAGE IS
PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
CONTAINING THE CONTOUR PLOT IS PRODUCED.
THREED IS CALLED TO DRAW A PLOT AND
PWRZT IS CALLED THREE TIMES TO LABEL THE
DRAW AXIS LINES. TPWRZT NEXT CALLS PWRZT
TO LABEL THE AXES FOR A THREE SPACE PLOT.
C
C REQUIRED ROUTINES
THREED

DIMENSION EYE(3)
DATA EYE(1), EYE(2), EYE(3) /3.5, 3.0, 5.0/
C
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C SELECT NORMALIZATION TRANS NUMBER 0 AND ESTABLISH THE MAPPING OF
C THREE SPACE COORDINATES ONTO THE GRAPHICS DEVICE COORDINATE SYSTEM.
CALL GSELNT (0)
CALL SET3 (.1,.9,.1,.9,0,.1,.0,.1,.0,.1,.0,.1,.EYE)
C
C THE FOLLOWING THREE CALLS TO LINE3 DRAW THE THREE SPACE AXES
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 LABEL EACH AXES, THE FIRST 3 PARAMETERS AND ICNT DO THE POSITIONING.
C LINE AND ITOP DETERMINE THE DIRECTION AND PLANE OF THE CHARACTERS.
ICNT = 0
LINE = 2
ITOP = 3
CALL PWRZT (0.,.5,.1,'V-AXIS',6,30,LIN,ITOP,ICNT)
LINE = -1
ITOP = 3
CALL PWRZT (.5,0.,.1,'U-AXIS',6,30,LIN,ITOP,ICNT)
LINE = -2
ITOP = -2
CALL PWRZT (0.,.1,.5,'Z-AXIS',6,30,LIN,ITOP,ICNT)
LINE = 2
ITOP = -1
ICNT = -1
CALL PWRZT (.5,2,0.,'DEMONSTRATION OF PWRZT WITH THREED',
1 34,30,LIN,ITOP,ICNT)
C
C A CALL TO FRAME INDICATES THAT THE PICTURE IS COMPLETE
CALL FRAME
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDWK(1)
CALL GCWK(1)
CALL GCGLK
WRITE (6,1001)
1001 FORMAT (' PWRZT TEST SUCCESSFUL',24X,
1 'SEE PLOT TO VERIFY PERFORMANCE')
STOP
END.

April 1986 -76- EXAMPLES
PROGRAM TSRFAC
C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF SRFACE
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE CONTOUR PLOT IS PRODUCED.
C
C REQUIRED ROUTINES NONE
C
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, ANGV CONTAINS THE ANGLE IN DEGREES FROM
C 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),ANGH,ANGV/
1 -8.0,-6.0,3.0,0.0,0.0,0.0,45.,15./
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM
C 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA CX/.5/,CY/.9/
C
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 1./((X+.1)**2+Y**2+.09))*.25
10 CONTINUE
20 CONTINUE
C
C OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
A DEMONSTRATION OF EZSRFC IS PRESENTED BELOW BUT IS NOT EXECUTED.
C SET TEXT ALIGNMENT TO CENTER THE STRING AT THE STRING CENTER
C AND IN THE VERTICAL CENTER; SET CHARACTER HEIGHT; PLOT CHARACTERS
C CALL GSTXAL(2,3)
C CALL GSCHH(.016)
C CALL GTX(CX,CY,'DEMONSTRATION PLOT FOR EZSRFC ENTRY OF SRFACE')
C CALL EZSRFC (Z,21,25,ANGH,ANGV,WORK)
C
C SRFACE DEMO
C SET TEXT ALIGNMENT TO CENTER THE STRING AT THE STRING CENTER
C AND IN THE VERTICAL CENTER; SET CHARACTER HEIGHT; PLOT CHARACTERS
C CALL GSTXAL(2,3)
C CALL GSCHH(.016)
C CALL GTX(CX,CY,'DEMONSTRATION PLOT FOR SRFACE ENTRY OF SRFACE')
C CALL SRFACE (XX,YY,Z,WORK,21,21,25,0.)
C
C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDWK(1)
CALL GCLWK(1)
CALL GCLKS
WRITE (6,1001)
1001 FORMAT (' SRFACE SUCCESSFUL',24X,'SEE PLOT TO CHECK PERFORMANCE')
STOP
END
DEMONSTRATION PLOT FOR SURFACE ENTRY OF SURFACE
PROGRAM TSTRML

C PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF STRMLN
C
C I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS
C PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME
C CONTAINING THE SAMPLE PLOT IS PRODUCED.
C ROUTINE TSTRML CALLS ROUTINE STRMLN TO
C PRODUCE A PLOT REPRESENTING THE FLOW AND
C MAGNITUDE OF A VECTOR FIELD.
C
C REQUIRED ROUTINES GRIDAL (BECAUSE PERIM IS CALLED BY TSTRML)
C
C REAL U(21,25),V(21,25),WRK(1050)
C
C SPECIFY COORDINATES FOR PLOT TITLES. THE COORDINATES RANGE FROM
C 0.0 TO 1.0 AND DEFINE THE CENTER OF THE TITLE STRING.
DATA TX/.5/,TY/.9765/
C
C SET DIMENSIONS
DATA NH,NV/21,25/
C
C SPECIFY HORIZONTAL AND VERTICAL VECTOR COMPONENTS U AND V ON
C THE RECTANGULAR GRID
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
OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION, SELECT NORMALIZATION TRANS
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)
CALL GSELNT(0)
C CALL WTSTR FOR STRMLN PLOT TITLE
CALL WTSTR (TX,TY,'DEMONSTRATION PLOT FOR STRMLN',2,0,0)
C
C DEFINE NORMALIZATION TRANSFORMATION 1, AND SET UP LOG SCALING
C AND DRAW PERIMETER
CALL SET(0.1, 0.9, 0.1, 0.9,1.0, 21., 1.0, 25.,1)
CALL PERIM(1,0,1,0)
C
C CALL STRMLN FOR VECTOR FIELD STREAMLINES PLOT
CALL STRMLN (U,V,WRK,NH,NV,1,IER)
CALL FRAME
C
DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDWIK(1)
CALL GCLKS
WRITE (6,1001)
1001 FORMAT (' STRMLN TEST SUCCESSFUL',24X,
1 'SEE PLOT TO VERIFY PERFORMANCE')
STOP
END
DEMONSTRATION PLOT FOR STRMLN
PROGRAM TTHREE

PURPOSE ..

PROVIDES A SIMPLE DEMONSTRATION OF THREED

I/O

IF THE TEST IS SUCCESSFUL, A MESSAGE IS

REQUIRED ROUTINES

PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME

CONTAINING THE CONTOUR PLOT IS PRODUCED.

NONE

THE VALUES RXA, RXB, RYA, AND RYB DEFINE THE PORTION OF ADDRESS

SPACE TO BE USED IN MAKING THE PLOT. UC, UD, VC, VD, WC, WD DEFINE

A VOLUME IN USER COORDINATES WHICH IS TO BE MAPPED ONTO THE VIEWING

SURFACE AS SPECIFIED BY RXA, RXB, RYA, AND RYB.

REAL EYE(3), X(31), Y(31), Z(31)

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/, PI/3.1415926535898/

OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION

CALL GOPKS(6)

CALL GOPWK(1,2,1)

CALL GACWK(1)

CALL SET3 TO ESTABLISH A MAPPING BETWEEN THE PLOTTER ADDRESSES

AND THE USER'S VOLUME. CALL PERIM TO DRAW PERIMETERS

CALL SET3(RXA, RXB, RYA, RYB, UC, UD, VC, VD, WC, WD, EYE)

CALL PERIM3(2, 5, 1, 10, 1, -1.)

CALL PERIM3(4, 2, 1, 1, 2, -1.)

CALL PERIM3(2, 10, 4, 5, 3, -1.)

DRAW LATITUDINAL LINES ON THE SPHERE OF RADIUS 1 AND CENTER (0., 0., 0.)

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

DRAW LONGITUDINAL LINES ON THE SPHERE

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

Y(J) = SIN(TUETA)*CP.

Z(J) = SP

40 CONTINUE

CALL CURVE3(X, Y, Z, 31)

30 CONTINUE

CALL WTSTR FOR THREED PLOT TITLE

CALL WTSTR(TX, TY, 'DEMONSTRATION PLOT FOR ROUTINE THREED', 2, 0, 0)

CALL FRAME

WRITE(6, 1001)

1001 FORMAT(' THREED SUCCESSFUL', 24X, 'SEE PLOT TO CHECK PERFORMANCE')

DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS

CALL GDWK(1)

CALL GCLKS

STOP

END
DEMONSTRATION PLOT FOR ROUTINE THREED
PROGRAM TVELVC

PURPOSE PROVIDES A SIMPLE DEMONSTRATION OF SUBROUTINES VELVCT AND EZVEC.

I/O IF THE TEST IS SUCCESSFUL, A MESSAGE IS PRINTED ON UNIT 6 TO THAT EFFECT AND 1 FRAME CONTAINING THE SAMPLE PLOT IS PRODUCED.

REQUIRED ROUTINES NONE

DIMENSION U(21,25), V(21,25)

C SPECIFY COORDS FOR PLOT TITLES AND SOME VELVCT ARGUMENTS
DATA CX/.09/,CY/.97/
DATA FLO/0./,HI/0./,NSET/0./,LENGTH/0./,ISPV/0./,SPV/0./

C SPECIFY VELOCITY FIELD FUNCTIONS U AND V
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.
        DZDY = 1.-2.*Y/((X-.10)**2+Y**2+.09)**2 + 1.
        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 OPEN GKS, OPEN AND ACTIVATE THE WORKSTATION
CALL GOPKS(6)
CALL GOPWK(1,2,1)
CALL GACWK(1)

A SAMPLE CALL FOR EZVEC IS PRESENTED BELOW BUT NOT CALLED:
CALL EZVEC (U,V,M,N)

CALL VELVCT FOR VELOCITY FIELD PLOT
CALL VELVCT (U,M,V,M,M,N,FLO,HI,NSET,LENGTH,ISPV,SPV)

CALL WTSTR FOR VELVCT PLOT TITLE
CALL GQCNTN(IERR,ICN)
CALL GSELNT(0)
CALL WTSTR (CX,CY,'DEMONSTRATION PLOT FOR VELVCT',2,0,-1)
CALL FRAME
WRITE (6,1001)
1001 FORMAT(' VELVCT SUCCESSFUL',24X,'SEE PLOTS TO CHECK PERFORMANCE')

C DEACTIVATE AND CLOSE THE WORKSTATION, CLOSE GKS
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLKS
STOP
END

April 1986
DEMONSTRATION PLOT FOR VELVCT

April 1986

0.371E+01
MAXIMUM VECTOR

EXAMPLES
In this section we present some examples of more sophisticated use of the Graphics Utilities. A complete program is provided along with the plots produced from the program. The examples currently in this program are:

1.) Three curves on the same plot using AUTOGRAPH with a user-defined dash pattern.
2.) A CONREC plot using a polar coordinate transformation.
3.) A CONREC plot overlaid on an EZMAP plot.
4.) Another CONREC overlaid on EZMAP.
5.) A CONREC plot with several user-defined parameters and a modified background using GRIDAL.
6.) 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 is in order.

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 currently implemented with this option are: the CONREC family, the CONRAN family, ISOSRF, STRMLN, and VELVCT. The way the transformation option is implemented is to include arithmetic statement functions in the selected utilities which 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 named FX and FY (for converting X-coordinates and Y-coordinates respectively.) So, initially the statement functions FX and FY are simply

\[
\begin{align*}
FX(X) &= X \\
FY(Y) &= Y \\
\end{align*}
\]

and these are the functions which must be replaced in the non-default case.

The range of the arguments to the statement functions is dependent on the routine in use. The examples supplied in the example program which follows illustrate this (the coordinate spaces for CONREC and VELVCT being FLOAT(1) to FLOAT(N) by FLOAT(1) to FLOAT(M) where the input array is dimensioned for N x M.)
Following is a list of file names and the subroutines in the given file which 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, some special considerations pertain to superimposing VELVCT plots onto other plots (regarding the length of the vectors.) The write-up for VELVCT contains details on this.
PROGRAM BNCHMK

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:

1.) Modify the OPEN to the EZMAP dataset that appears in the main program.

2.) Modify the OPEN to the file of random numbers RANFDAT in subroutine RAND.

The utilities AUTOGRPH, 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(l,5),B(l,l))
COMMON /TRANS/ MA,NA,ITRANS,A1,A2,D1,D2,ALNMN,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')

Open GKS, open workstation, activate workstation.

CALL GOPKS (6)
CALL GOPWK (1, 2, 1)
CALL GACWK (1)
ZERO = CHAR(0)

Generate the input array.

CALL GENARA(B,A,M,N)

Frame 1 -- three curves on same plot using AUTOGRAPH.

CALL FRNUM(1)
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)
160 CONTINUE
CALL FRAME

Frame 2 -- CONREC plot with polar coordinate transformation.

ITRANS = 1
D1 = 20.
MA = 30
NA = 70
CALL GENARA(B,A,MA,NA+12)
CALL FRNUM(2)
XMIN = -60.
XMAX = 60.
YMIN = -80.
YMAX = 80.
CALL SET(.1,.9,.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 = 1.25*(D1+D2)
DELY = 1.50*(D1+D2)
CALL SET(.05,.95,-.05,.95,-D1,DELY,-DELY,DELY,1)
CALL CONREC (A(1,4),MA,MA,NA,0.,0.,0.,1,-1,0)
CALL FRAME

Frame 3 -- CONREC overlaid on EZMAP.

MA = 55
NA = 45
CALL GENARA(B,A,MA,NA)
ITRANS = 2
CALL FRNUM(3)
CALL SUPMAP (2,40.,-90.,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 overlaid on EZMAP.

MA = 55
NA = 45
CALL GENARA(B,A,MA,NA)
ITRANS = 2
CALL FRNUM(4)
CALL SUPMAP (3,45.,-100.,45.,50.,-130.,20.,-75.,2,10,3,1,IERs)
ALNMN = -120.
ALNMX = -75.
ALTMN = 25.
ALTMX = 50.
CALL CONREC (A,MA,MA,NA,0.,0.,0.,1,-1,1)
CALL FRAME

Frame 5 -- CONREC with modified background.

ITRANS = 0
CALL FRNUM(5)
CALL GENARA(B,A,40,40)
CALL SET(.1,.9,.8,.9,1.,1.,1.,1.,-2.,2.,1)
CALL LABMOD ('(F3.0)' ,'(F3.0)' ,3,3,0,0,0,0,0,0)
CALL PERIML(2,10,4,5)
CALL CONREC(A(1,1),40,40,40,-1.,1.,1.,5,-1,-1,-682)
CALL FRAME

Frame 6 -- VELVCT overlaid on EZMAP.

CALL GENARA(B,A,60,60)
ITRANS = 2
CALL FRNUM(6)
CALL SUPMAP (3,45.,-100.,45.,50.,-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

SUBROUTINE FRNUM(ISTART)

This subroutine plots a frame number in the upper right-hand corner of a plot each time it is called. If ISTART.EQ.0, then NUM is bumped by 1 and the value of NUM is plotted; if ISTART.GT.0, then NUM is set to ISTART and the value of NUM is plotted.

SAVE NUM
CHARACTER*5 CHARS
DATA NUM/0/
IF (ISTART .EQ. 0) THEN
   NUM = NUM+1
ELSE IF (ISTART .GT. 0) THEN
   NUM = ISTART
ELSE
   WRITE(6,100) ISTART
   STOP
ENDIF
WRITE (CHARS,'(I5)') NUM
CALL GQCNTN(IER,ICN)
CALL GSELNT(0)
CALL WTSTR(.94,.98,CHARS,1,0,0)
CALL GSELNT(ICN)
RETURN

100 FORMAT(' FRNUM-ISTART OUT OF RANGE',I10)

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), (D2,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, ALMNX, 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.

CONTINUE
FX = X
RETURN

Polar coordinate transformation.

CONTINUE
FX=(D1+D2*(X-1.)/(FLOAT(MA)-1.))*COS(A1+A2*(Y-1.)/(FLOAT(NA)-1.))
RETURN

EZMAP overlaying.

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)
FX = FXLON
RETURN

FUNCTION FY(X,Y)

COMMON /TRANS/ MA,NA,ITRANS,A1,A2,D1,D2,ALNMN,ALNMX,ALTMN,ALTMX

The function FY behaves in an analogous manner to FX as described above.

GOTO (10,20,30) ITRANS+1

The identity transformation.

CONTINUE
FY = Y
RETURN

Polar coordinate transformation.

CONTINUE
FY=(D1+D2*(X-1.)/(FLOAT(MA)-1.))*SIN(A1+A2*(Y-1.)/(FLOAT(NA)-1.))
RETURN

EZMAP overlaying.

CONTINUE
XLON = ALNMN + (X-1.)*(ALNMX-ALNMN)/(FLOAT(MA)-1.)
YLAT = ALTMN + (Y-1.)*(ALTMX-ALTMN)/(FLOAT(NA)-1.)
CALL MAPTRN(YLAT, XLON, XDUM, FYLAT)
FY = FYLAT
RETURN
END

SUBROUTINE GENARA (A,B,ID,JD)

April 1986
This subroutine generates a smooth array in output array B. The array is dependent on the random number function RAND. 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.

```
DIMENSION A(ID,JD), B(ID,JD)

DO 1 I=1,ID
  DO 1 J=1,JD
    1 A(I,J)=0.

NN=(ID+JD)/10
AA=1.
DI=ID-4
DJ=JD-4
DD=MAX0(ID,JD)

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=MAX0(IE,JE)
         A(I,J)=A(I,J)+AA*(.8**EE)
      3 CONTINUE
   4 CONTINUE
   5 CONTINUE

IF (AA.NE.1.) GO TO 6
   AA=-1.
   GO TO 2

6 DO 8 J=1,JD
   JM1=MAX0(1,J-1)
   JP1=MIN0(JD,J+1)
   DO 7 I=1,ID
      IM1=MAX0(1,I-1)
      IP1=MIN0(ID,I+1)
            +A(IM1,JM1)+A(IP1,JM1)+A(IM1,JP1)+A(IP1,JP1))/16.
   7 CONTINUE
   8 CONTINUE
}
RETURN
END
```

FUNCTION RAND()

This function is used to produce random numbers for the GENARA calls. The random numbers are read from file RANFDAT, and in this way consistency is maintained across machines (the results on one machine should be the same as on another.) If consistency is not required, the function RAND can be replaced by a local random number generator. RANFDAT contains 2000 random numbers in 250 card-image lines in format 8F10.8.

```
DIMENSION A(2000)
DATA ICNT/0/

ICNT = ICNT+1
}
```

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
C
RAND = A(ICNT)
RETURN
C
500 FORMAT(' RAND--ERROR IN READ')
501 FORMAT(8F10.8)
C
END
CONTOUR FROM -1.4000 TO 1.8000 CONTOUR INTERVAL OF 0.20000
April 1986
PT(3.3) = -0.47078
EXAMPLES
-97-
CONTOUR FROM -1.0000 TO 1.0000 CONTOUR INTERVAL OF 0.50000
PT(3.3) = 0.13882
April 1986
The following is an example which exercises the five basic GKS output primitives. On translation, the cell array generated by the GCA call is represented at the minimal level (i.e. a boundary is drawn around the array.) With a more sophisticated display, a checkerboard pattern would be drawn. A sample plot follows the code listing.
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, 4.25, 5.26, 6.15, 6.50/
DATA CIRY / 8.46, 8.98, 8.80, 8.01, 6.9, 6.20, 6.02, 6.54, 7.50/
DATA ISZ/0/

CALL GOPKS (6,ISZ)
CALL GOPWK (1, 2, 1)
CALL GACWK (1)

Open GKS w/ logical unit 6 for errors.
Open a wkstn of type 1, assigning 1 for its id, 2 for connection id.
Activate the workstation.

CALL GSWN (1, -10.0, 10.0, -10.0, 10.0)
CALL GSVP (1, 0.1, 0.9, 0.1, 0.9)
CALL GSELNT (1)

Define normalization transformation 1. Window in World Coordinates.
Viewport in Normalized Device Coords.
Select just-defined transformation.

CALL GPL (9, ZZX, ZZYL)
"+
CALL GSMK (2)
CALL GPM (9, ZZX, ZZYM)
solid
CALL GSFAIS (1)
CALL GFA (9, CIRX, CIRY)

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

April 1986

-103-

EXAMPLES
```
ICELLS (IX,IY) = 0
ENDIF
CONTINUE
CONTINUE

Draw the checkerboard w/ CELL ARRAY.
CALL GCA (2.,1.5,8.,4.0, 24, 12, 1, 1, 24, 12, ICELLS)

CALL GSCHH (0.4)  Set chr hgt to 2% of screen (.02*20).
Set alignment to center string on posn.
CALL GSTXAL (2, 3)
XPOS = 0.0
YPOS = -5.0

CALL GTX (XPOS, YPOS, 'Example string, centered in the center')  Draw the text string.
Flush metacode buffer.
CALL GCLRWK(1,1)  Deactivate the workstation.
CALL GDAWK (1)  Close the workstation.
CALL GCLWK (1)  Close GKS
CALL GCLKS
STOP
END
Example string, centered in the center
The following example produces one simple plot. If an NCAR CGM is created, a metafile dump is listed in hex and in octal in the code documentation. This dump can be used to check if a proper metafile is being created. The sample plot appears after the code listing.
This program produces a small plotting instruction stream. The primary purpose of this program is to test a local implementation of the NCAR GKS 0A package, including PLOTIT of SPPS. Only a trivial subset of the package is used in this minimal test. Execution of this program using NCAR's GKS package will produce a CGM. Hex and octal dumps of this CGM follow:

In HEX:

Record 1:
001E 3400 0021 0000 1022 8001 104B 0A4E
4341 525F 474B 5330 4100 1166 0001 FFFF
0000 0000 0000 0000 0000 0000 0000 0000

(86 rows of zeros as the above line.)
0000 0000 0000 0000 0000 0000 0000 0000

Record 2:
005C 3800 0061 0000 0080 30C2 0001 4034
0000 0000 7FFF 0000 7FFF 7FFF 0000 7FFF
0000 0000 524C 0002 0003 0000 0000 0000
0000 51E2 02D5 5208 0000 02D5 02D5 0000
409C 3FFF 3FFF 0000 154E 4341 5220 474B
5320 5061 636B 6167 6520 5465 7374 00A0
0000 0000 0000 0000 0000 0000 0000 0000

(82 rows of zeros as in the above line.)
0000 0000 0000 0000 0000 0000 0000 0000

Record 3:
0002 3200 0040 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000

(87 rows of zeros as in the previous line.)
0000 0000 0000 0000 0000 0000 0000 0000
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:
\[
\begin{array}{cccccccccccccccc}
000 & 036 & 064 & 000 & 000 & 041 & 000 & 000 & 020 & 042 & 200 & 001 & 020 & 113 & 012 & 116 \\
103 & 101 & 122 & 137 & 107 & 113 & 123 & 060 & 101 & 000 & 021 & 146 & 000 & 001 & 377 & 377 \\
000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 \\
\end{array}
\]

\[
\text{(86 rows of zeros as the above line.)}
\]

Record 2:
\[
\begin{array}{cccccccccccccccc}
000 & 134 & 070 & 000 & 000 & 141 & 000 & 000 & 000 & 200 & 060 & 302 & 000 & 001 & 100 & 064 \\
000 & 000 & 000 & 000 & 177 & 377 & 000 & 000 & 177 & 377 & 177 & 377 & 000 & 000 & 177 & 377 \\
000 & 000 & 000 & 000 & 122 & 114 & 000 & 002 & 000 & 003 & 000 & 000 & 000 & 000 & 000 & 000 \\
000 & 000 & 121 & 342 & 002 & 325 & 122 & 010 & 000 & 000 & 022 & 325 & 002 & 325 & 000 & 000 \\
100 & 234 & 077 & 377 & 077 & 377 & 000 & 000 & 025 & 116 & 103 & 101 & 122 & 040 & 107 & 113 \\
123 & 040 & 120 & 141 & 143 & 153 & 141 & 147 & 145 & 040 & 124 & 145 & 163 & 164 & 000 & 240 \\
000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 \\
\end{array}
\]

\[
\text{(82 rows of zeros as in the above line.)}
\]

Record 3:
\[
\begin{array}{cccccccccccccccc}
000 & 002 & 062 & 000 & 000 & 100 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 \\
000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 & 000 \\
\end{array}
\]

\[
\text{(87 rows of zeros as in the previous line.)}
\]

April 1986

EXAMPLES
C record due to the fact that the metafile output buffer has
C not been cleared from previous writes (clearing is not
C necessary since the record contains a count of how many
C bytes are significant in the record.)
C
DATA N/32767/
C
C OPEN GKS, OPEN WORKSTATION OF TYPE 1, ACTIVATE WORKSTATION
C
CALL GOPKS (6,ISIZ)
CALL GOPWK (1, 2, 1)
CALL GACWK (1)
C
C TRIVIAL FRAME
C
CALL SET(0.,1.,0.,0.,0.,1.,0.,1.,0.,1.)
CALL PLOTIT (0,0,0)
CALL PLOTIT (N,0,1)
CALL PLOTIT (N,N,1)
CALL PLOTIT (0,N,1)
CALL PLOTIT (0,0,1)
CALL WTSTR(.5,.5,'NCAR GKS Package Test',3,0,0)
CALL FRAME
C
C DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS.
C
CALL GDAWK (1)
CALL GCLWK (1)
CALL GCLKS
STOP
END

April 1986
The following example can be used to test SPPS. The non-plotting as well as plotting entries are tested. Twenty four frames are produced. Examples of the plots appear after the code listing.
COMMON /BLOCK1/MESG,IDUMMY(500)

Use the SPPS entry to open GKS.

CALL OPNGKS

Check the non-plotting entries.

SET and GETSET (SET is also tested below.)

CALL SET( 0., 1., 0., 1., 10., 100., 1 )
CALL GETSET(PXA,PXB, PYA, PYB, XC, XD, YC, YD, LTYPE)
IF (PXA NE.0. OR. PXB NE.1. OR.
- PYA NE.0. OR. PYB NE.1. 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)
STOP
210 CONTINUE

C FL2INT

CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL FL2INT(0.,0.,MX,MY)
IF (MX NE.0. OR. MY NE.0) WRITE(MESG,530)
IF (MX NE.0. OR. MY NE.0) GO TO 220
CALL FL2INT(1.,1.,MX,MY)
IF (MX NE.32767. OR. MY NE.32767) WRITE(MESG,530)
IF (MX NE.32767 . OR. MY NE.32767) GO TO 220
CALL FL2INT(.5,.5, MX, MY)
IF (MX NE.16383. OR. MY NE.16383) WRITE(MESG,530)
IF (MX NE.16383. OR. MY NE.16383) GO TO 220
WRITE(MESG,520)
220 CONTINUE

C SETI and GETSI

CALL SETI(13,5)
CALL GETSI(LX,LY)
IF (LX NE.13 . OR. LY NE.5) WRITE(MESG,550)
IF (LX NE.13 . OR. LY NE.5) GO TO 230
WRITE(MESG,540)
CALL SETI(10,10)
230 CONTINUE

C FRSTPT and MXMY

CALL FRSTPT(.5,.5)
CALL MXMY(MX,MY)
IF (MX NE.512. OR. MY NE.512) WRITE(MESG,570)
IF (MX NE.512 . OR. MY NE.512) GO TO 240
WRITE(MESG,560)
240 CONTINUE
250 CONTINUE

C Test plotting entries.

CALL TPWRIT
CALL TPLOTI
CALL TLINE
CALL TCURVE
CALL TVECTO
CALL TSET
CALL TPERIM
CALL THALFA
CALL TTICK4
CALL TGRID
CALL TGRIDL
CALL TPERML
CALL TLABMO
CALL TPNTS
CALL TGRIDA

C Use the SPPS entry to close GKS.
C
CALL CLSGKS
STOP

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')

END

SUBROUTINE BOX
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

SUBROUTINE FRNUM(ISTART)
C
C This subroutine writes a frame number in the upper right hand
C portion of the frame each time it is called. If ISTART.EQ.0,
C NUM is bumped by 1 before it is written. If ISTART.EQ.0, NUM
C is set to ISTART.
C
COMMON /BLOCK1/MESG,IDUMMY(500)
CHARACTER*5 CHRS
DATA NUM/0/
CALL GETSET(IV1,IV2,IV3,IV4,IW1,IW2,IW3,-IW4,ILG)
CALL SET(0O.,.,0.,1.,0.,1.,O.,.,l) ,)
IF (ISTART.GT.0) GO TO 10
IF (ISTART.EQ.0) GO TO 20
WRITE(MESG,100) ISTART
STOP
20 CONTINUE
NUM = NUM+1
GO TO 30
10 CONTINUE
NUM = ISTART
30 CONTINUE
WRITE(CHRS,223) NUM
223 FORMAT(I5)
CALL PWRIT(.93,.96,CHRS,5,20,0,0)
CALL SET(IV1,IV2,IV3,IV4,IW1,IW2,IW3,IW4,ILG)
RETURN

C
100 FORMAT(' FRNUM-ISTART OUT OF RANGE',I10)
C
April 1986 EXAMPLES-118-
SUBROUTINE TCURVE
  COMMON /BLOCK1/MESGX(51),Y(51),IDUMMY(398)
  CALL BOX
  CALL FRNUM(0)
  CALL SET(0.,1.,0.,1.,0.,1.,0.,.3,1)
  CALL PWRIT(0.5, .274,
    + 'DEMONSTRATION PLOT FOR CURVE',28,20,0,0)
  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
SUBROUTINE TGRID
  COMMON /BLOCK1/ MESG,IDUM(500)
  CALL FRNUM(0)
  CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
  CALL PWRIT(0.5, 935./1024.,
    + 'DEMONSTRATION PLOT FOR GRID',27,20,0,0)
  CALL SET(.2,.8,.2,.8,.2,.8,.2,.8,1)
  CALL GRID(5,2,6,3)
  CALL FRAME
  WRITE(MESG,610)
  610 FORMAT(' TGRID EXITED--SEE PLOT TO VERIFY PERFORMANCE')
  RETURN
END
SUBROUTINE TGRIDA
  COMMON /BLOCK1/ MESG,IDUMMY(500)
  CALL SET(.2, .8,.2,8,.2,8,.2,8,1)
  DO 100 I=1,9
    CALL FRNUM(0)
    IGPH = I-1
    IF (I .GT. 3) IGPH = IGPH+1
    IF (I .GT. 6) IGPH = IGPH+1
    WRITE(CHRS,800) IGPH
    800 FORMAT(I2)
    CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
    CALL PWRIT(575./1024., 895./1024., CHRS,2,20,0,'-1')
    CALL PWRIT(0.5, 895./1024.,
      + 'IGPH = ' 7,20,0,0)
    CALL PWRIT(0.5, 935./1024. ,
      + 'DEMONSTRATION PLOT FOR GRIDAL',29,20,0,0)
    CALL SET(.2,.8,.2,.8,.2,.8,.2,.8,1)
    CALL GRIDAL(5,2,6,3,1,1,IGPH,.2, .2)
    CALL FRAME
  100 CONTINUE
  WRITE(MESG, 600)
  600 FORMAT(' TGRIDA EXITED--SEE PLOT TO VERIFY PERFORMANCE')
  RETURN
END
SUBROUTINE TGRIDL
  COMMON /BLOCK1/ MESG,IDUM(500)
  CALL FRNUM(0)
  CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
  CALL PWRIT(0.5, 935./1024.,
    + 'DEMONSTRATION PLOT FOR GRIDL',28,20,0,0)
CALL SET(.2,.8,.2,.8,0.,1.,0.,1.,1)
CALL GRIDL(5,2,6,3)
CALL FRAME
WRITE(MESG, 620)
620 FORMAT(' TGRIDL EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END

SUBROUTINE THALFA
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRITE(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR HALFA',29,20,0,0)
CALL SET(.2,.8,.2,.8,0.,1.,0.,1.,1)
CALL FRNUM(0)
CALL HALFA(5,2,6,3,0.,0.,0,0)
CALL FRAME
WRITE(MESG, 680)
680 FORMAT(' THALFA EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END

SUBROUTINE TLABMO
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL FRNUM(0)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRITE(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR LABMOD',29,20,0,0)
CALL SET(.2,.8,.2,.8,0.,1.,0.,1.,1)
CALL LABMOD('(E10.2)', '(F4.2)',10,4,15,15,0,0,0)
CALL HALFA(5,2,6,3,0.,0.,0,0)
CALL FRAME
WRITE(MESG, 690)
690 FORMAT(' TLABMO EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END

SUBROUTINE TLINE
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL FRNUM(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 FRAME
WRITE(MESG,700)
700 FORMAT(' TLINE EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END

SUBROUTINE TPERIM
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRITE(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR PERIM',28,20,0,0)
CALL SET(.2,.8,.2,.8,0.,1.,0.,1.,1)
CALL FRNUM(0)
CALL PERIM(5,2,6,3)
CALL FRAME
WRITE(MESG, 660)
660 FORMAT(' TPERIM EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END

SUBROUTINE TPERML
COMMON /BLOCK1/ MESSG,IDUMMY(500)
CALL FRNUM(0)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRIT(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR PERIML',29,20,0,0)
CALL SET(.2,.8,.2,.8,0.,1.,0.,1.,1)
CALL PERIML(5,2,6,3)
CALL FRAME
WRITE(MESG,670)
670 FORMAT(' TPERML EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
SUBROUTINE TPLOTI
COMMON /BLOCK1/ MESSG,IDUMMY(500)
CALL FRNUM(0)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRIT(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR PLOTIT',29,20,0,0)
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
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 PWRIT(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR POINTS',29,20,0,0)
DO 130 I=1,9
 X(I) = 1.*I
 Y(I) = .50
130 CONTINUE
ICH = ICHAR('X')
CALL POINTS(X,Y,9,ICH,1)
CALL FRAME
WRITE(MESG,730)
730 FORMAT(' TPNTS EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END
SUBROUTINE TPOINT
COMMON /BLOCK1/ MESSG,IDUMMY(500)
CALL FRNUM(0)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1)
CALL PWRIT(0.5, 935./1024.,
+ 'DEMONSTRATION PLOT FOR POINT',28,20,0,0)
DO 120 I=1,512
 IX = I*2
 IY = 1024-IY
 CALL POINT(FLOAT(IX)/1024.,FLOAT(IY)/1024.)
 IY = 1024-IY
 CALL POINT(FLOAT(IX)/1024.,FLOAT(IY)/1024.)
120 CONTINUE
CALL FRAME
WRITE(MESG,720)
720 FORMAT(' TPOINT EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
April 1986 -121- EXAMPLES
SUBROUTINE TPWRIT
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL FRNUM(O)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PWRIT(0.5, 0.5,'DEMONSTRATION PLOT FOR PWRIT',28,30,0,0)
CALL FRAME
WRITE(MESG,740)
740 FORMAT(' TPWRIT EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
SUBROUTINE TSET
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL FRNUM(O)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PWRIT(0.5, .975.
+ 'DEMONSTRATION PLOT FOR SET',26,25,0,0)
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
SUBROUTINE TTICK4
COMMON /BLOCK1/ MESG,IDUMMY(500)
CALL FRNUM(O)
CALL SET(0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PWRIT(0.5, 935./1024.
+ 'DEMONSTRATION PLOT FOR TICK4',28,20,0,0)
CALL SET(.2,.8,.2,.8,0.,1.,0.,1.,1.)
CALL TICK4(150,50,150,50)
CALL PERIM(5,2,6,3)
CALL FRAME
CALL TICK4(12,8,12,8)
WRITE(MESG,750)
750 FORMAT(' TTICK4 EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
SUBROUTINE TVECTO
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)
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 PWRIT(0.5, .274,
+ 'DEMONSTRATION PLOT FOR VECTOR',29,20,0,0)
CALL FRAME
WRITE(MESG,630)
630 FORMAT(' TVECTO EXITED--SEE PLOT TO VERIFY PERFORMANCE')
RETURN
END

BLOCKDATA
COMMON /BLOCK1/MESG, IDUMMY(500)

C
C Set output unit for messages.
C
DATA MESG/6/
END
DEMONSTRATION PLOT FOR PWRIT
DEMONSTRATION PLOT FOR PLOTIT

April 1986

-127-

EXAMPLES
DEMONSTRATION PLOT FOR LINE

April 1986

EXAMPLES
DEMONSTRATION PLOT FOR CURVE
DEMONSTRATION PLOT FOR VECTOR
DEMONSTRATION PLOT FOR PERIM
DEMONSTRATION PLOT FOR HALIFAX

April 1986

EXAMPLES
DEMONSTRATION PLOT FOR TICK4
DEMONSTRATION PLOT FOR GRIDL

0.100E+01

0.833E+00

0.667E+00

0.500E+00

0.333E+00

0.167E+00

0.

April 1986

11

EXAMPLES-136--
DEMONSTRATION PLOT FOR PERIML
DEMONSTRATION PLOT FOR L'ABMOD
DEMONSTRATION PLOT FOR POINTS

April 1986

-140-

EXAMPLES
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 0
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 1

April 1986
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 2
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 4
DEMONSTRATION PLOT FOR GRIDAL

IGPH = 5
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 6
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 8

April 1986
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 9
DEMONSTRATION PLOT FOR GRIDAL
IGPH = 10

April 1986