# U.S. GEOLOGICAL SURVEY NATIONAL COMPUTER TECHNOLOGY MEETING: PROCEEDINGS, PHOENIX, ARIZONA, NOVEMBER 14-18, 1988


U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 90-4162

# U.S. GEOLOGICAL SURVEY NATIONAL COMPUTER TECHNOLOGY MEETING: PROCEEDINGS, PHOENIX, ARIZONA, NOVEMBER 14-18, 1988

By Barbara H. Balthrop and John E. Terry, editors

---

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 90-4162

USGS
*science for a changing world*

Click here to return to USGS Publications

Nashville, Tennessee

1991

# U.S. DEPARTMENT OF THE INTERIOR
# MANUEL LUJAN, JR., Secretary

# U.S. GEOLOGICAL SURVEY
## Dallas L. Peck, Director

---

*For additional information write to:*

*Coordinator, National Computer Technology Meeting Proceedings 1988*
*U.S. Geological Survey*
*300 W. Congress*
*Federal Building, FB-44*
*Tucson, Arizona 85701*

*Copies of this report can be purchased from:*

*U.S. Geological Survey*
*Books and Open-File Reports Section*
*Federal Center*
*Box 25425*
*Denver, Colorado 80225*

# FOREWORD

*The U.S. Geological Survey National Computer Technology Meetings (NCTM) are sponsored by the Water Resources Division and provide a forum for the presentation of technical papers and the sharing of ideas or experiences related to computer technology. This report serves as a proceedings of the meeting held in November, 1988 at the Crescent Hotel in Phoenix, Arizona. The meeting was attended by more than 200 technical and managerial people representing all Divisions of the U.S. Geological Survey.*

*Scientists in every Division of the U.S. Geological Survey rely heavily upon state-of-the-art computer technology (both hardware and software). Today the goals of each Division are pursued in an environment where high speed computers, distributed communications, distributed data bases, high technology input/output devices, and very sophisticated simulation tools are used regularly. Therefore, information transfer and the sharing of advances in technology are very important issues that must be addressed regularly.*

*This report contains complete papers and abstracts of papers that were presented at the 1988 NCTM. The report is divided into topical sections that reflect common areas of interest and application. In each section, papers are presented first followed by abstracts. For these proceedings, the publication of a complete paper or only an abstract was at the discretion of the author, although complete papers were encouraged.*

*Some papers presented at the 1988 NCTM are not published in these proceedings.*

John E. Terry

# CONTENTS

## CHAPTER E--DATA BASES AND AUTOMATED DATA HANDLING



## CHAPTER F--ELECTRONIC PUBLISHING/MANUSCRIPT PREPARATION

## CONVERSION FACTORS

| Multiply | By | To obtain |
|---|---|---|
| inch (in.) | 25.4 | millimeter |
| inch per day (in/d) | 25.4 | millimeter per year |
| foot (ft) | 0.3048 | meter |
| foot per day (ft/d) | 0.3048 | meter per day |
| foot per mile (ft/mi) | 0.1894 | meter per kilometer |
| cubic foot per second ($ft^3/s$) | 0.02832 | cubic meter per second |
| mile (mi) | 1.609 | kilmometer |
| square mile ($mi^2$) | 2.590 | square kilometer |
| mile per square mile ($mi/mi^2$) | 0.621 | kilometer per square kilometer |
| gallons per day (gal/d) | 0.003785 | cubic meter per day |
| million gallons (Mgal) | 3,785 | cubic meter |
| million gallons per day per square mile [$(Mgal/d)mi^2$] | 1,460 | cubic meter per day per square kilometer |

Temperature in degrees Celsius (°C) may be converted to degrees Fahrenheit (°F) as follows:

$$°F = 1.8 \times °C + 32$$

---

The use of brand, company, or trade names in this report is for indentification purposes only and does not constitute endorsement by the U.S. Geological Survey.

# CHAPTER A--USGS COMPUTER MANAGEMENT/ADMINISTRATION

# MAGNETIC-TAPE BACKUP AND ROUTINE MAINTENANCE PROCEDURES FOR A MINICOMPUTER SYSTEM OF THE U.S. GEOLOGICAL SURVEY

By John E. Owen

U.S. Geological Survey

## ABSTRACT

The Arkansas District of the U.S. Geological Survey relies on the dependability and efficiency of a minicomputer to process basic data, conduct interpretive studies, and to meet the needs of cooperators. Hydrologic data, processing programs, and user information on the minicomputer disk partitions are maintained in an accurate, logical, and secure manner. The procedures used to perform the daily, biweekly, and monthly backups as well as the procedures used for pulling, cleaning, and testing magnetic tapes are discussed in this paper. The schedules for performing magnetic-tape backup operations and routine maintenance procedures for the minicomputer also are discussed. A minimum of downtime and virtually no loss of data demonstrates the effectiveness of this operation in the Arkansas District.

## INTRODUCTION

The Arkansas District of the U.S. Geological Survey uses the PRIME minicomputer and its capability to respond to the needs of the district and its cooperators. The district maintains information in an accurate, logical, and secure manner with a minimum amount of downtime. This paper describes the steps, programs, and files used to perform daily (incremental), biweekly, and monthly magnetic-tape backups; the procedures used for pulling, cleaning, and testing magnetic tapes; and the schedule for performing routine maintenance of the minicomputer.

Cleanliness and controlled climate are essential to dependability of hardware and efficiency of the minicomputer. All tapes and spare disk packs are maintained in the controlled environment of the computer room or in an offsite, climatically-controlled vault. Limited access is allowed in these areas. All cleaning of the controlled area is accomplished by the computer operator.

## MAGNETIC-TAPE BACKUP PROCEDURES

Daily, biweekly, and monthly magnetic-tape backups are performed with the PRIMOS utility program (MAGSAV), which is monitored by a magnetic tape backup log and controlled by a locally written menu driven Command Procedure Language (CPL) program and executed by a Command Input (COMI) file. The Command Procedure Language makes use of such "high-level language" features as branching and argument transfer to simplify and automate long command

sequences and to allow decision making at the command level. The COMI file controls the flow of commands to the CPL, insuring that logical tape numbers will be standardized, providing expedient access in the event that file recovery becomes necessary. Index files also are maintained and stored for all backup tapes to quickly locate the tape that contains any on-line file that is lost or destroyed. The appropriate tape is then used to restore the damaged or destroyed file. This practice has virtually eliminated loss of data.

## Daily Backup

Daily backup tapes are maintained for 90 days. When the 90-day period has been exceeded, the tapes are pulled from the tape rack and the labels removed. The tapes are then processed through a tape cleaner before being returned to the computer room for use. All tapes are kept on a tape rack in a climatically-controlled room. Only tapes that have been in this environment for at least 24 hours are used. Tapes are not stored lying flat even when waiting to be mailed or shipped to offsite storage.

## Biweekly and Monthly Backup

Biweekly and monthly backups are performed on Sunday afternoons. This time has been selected because it does not interfere with normal operations and reduces the amount of down time. These backups, like the daily backups, are performed by the MAGSAV utility program, which is controlled by a CPL program. Biweekly backups are done on the first day of each pay period. The purpose of this date is to capture all data for the prior period. It is also easier for an employee to determine what day the computer will be unavailable. Also, long running models or other batch jobs have time to complete before the biweekly backups are done on Sunday. The biweekly back-up is executed with all system processes shut down and only the system is running. With priority access rights set for the system, every file is backed up. The same backup CPL program that is used for incrementals is used for the total backup and an index for each partition is created. Because of the large volume of records on disk, all tapes are created at a density of 6,250 bits per inch (BPI) which permits a faster completion time and minimizes the number of tapes that are used for backup. Because cold starts from tape require the command device tape to be written at 1,600 BPI, a spare disk pack is stored that can be loaded while maintenance is being performed on the damaged device. Biweekly tapes are stored on a tape rack, and are maintained for 6 months before being pulled, cleaned, and randomly tested before put back in use. These tapes are maintained in date and Master File Directory (MFD) order so that a complete set of backup tapes can be easily located by the operator if a file or directory restoration is necessary.

Monthly backups are done by the same procedure as the biweekly backups, but two copies of each partition are created. One copy stays on a rack designated for monthly backup tapes. The other goes to an offsite storage area that is under contract and maintained under strict access. The off-site tapes are kept in a climatically-controlled vault and can be delivered back to the computer room on request. Monthly tapes are held for 1 year

but, rotation of tapes within the cartridge seal is done every 90 days. Higher quality tapes are selected for monthly backups because of the importance of these master files. Cleaning and testing after 1 year is accomplished before tapes are released for reuse.

## ROUTINE MAINTENANCE PROCEDURES

Preventive maintenance on the minicomputer is performed by the field engineer on a monthly basis. At this time filters are cleaned or replaced and logs checked for disk errors. Because maintenance visits require the system to be shutdown, they are scheduled during lunch to minimize the amount of time that the machine is unavailable to users. Other routine maintenance operations are performed by the system operator at hours convenient to the user community because most routine maintenance only takes a few minutes. After the biweekly backups are completed, another COMI file is run which executes the PRIMOS disk maintenance utility program, FIX_DISK, which helps insure the integrity and accessibility of all files on the disk.

The minicomputer has the ability to support a remote systems console. When power fluctuations occur at night or on weekends, the computer can be cold started, FIX_DISK can be run to repair damaged files, and phantoms that run continuously can be started from the remote systems console. This convenience can save much travel time and also makes it possible to bring the system up when it might otherwise be down for several hours. A COMI file has been written to perform the FIX_DISK utility and a log is kept so that disk maintenance can be performed to minimize human error. Routine disk maintenance is completed after each biweekly backup in addition to unscheduled maintenance that is performed after power outages and so forth.

After disk maintenance is performed, the log is checked for any disk problem before restarting system phantoms. If a problem exists, the operator will determine if FIX_DISK is to be re-run or if the system administrator is to be notified of the problem. Once maintenance is determined to be complete and all disk files are usable, a COMI file is started to bring up system phantoms.

## SUMMARY

The Arkansas District of the U.S. Geological Survey relies on the dependability and efficiency of a minicomputer to process data, do interpretive studies, and meet the demands of its cooperators. With the use of the minicomputer software packages, locally written CPL programs, and COMI files, maintenance and operations of the minicomputer system are accomplished efficiently. A minimum of downtime and virtually no loss of data demonstrates the effectiveness of this operation in the Arkansas District.

## SELECTED REFERENCES


Alley, S.A., 1984, Magnetic tape user's guide: Natick, Massachusetts, PRIME Computer, Inc., chapter 9.

-----, 1984, Operator's guide to system backups: Natick, Massachusetts, PRIME Computer, Inc., chapters 1, 5, 6, 8, and 14.

Forbes, J., Landy, A., and Miles, C., 1986, Operator's guide to system commands: Natick, Massachusetts, PRIME Computer Inc., p. 2-40, 2-73, 2-9, and 2-10.

Froenlich, A.F., 1982, Managing the data center: Belmont, California, Wadsworth Inc., Lifetime Learning Publications, p. 19-21, 254.

Gaither, N., 1984, Production and operations management: New York, CBS College Publishing, Dryden Press, p. 710-724.

Gove, G.W., 1984, Operator's guide to file system maintenance: Natick, Massachusetts, PRIME Computer, Inc., chapter 3.

# THE DEVELOPMENT OF DISTRIBUTOR SOFTWARE FOR TRANSMITTING DOCUMENTS THROUGH A COMPUTER NETWORK

by
Steven J. Brady[1]

## ABSTRACT

The use of electronic mail by the Water Resources Division of the U.S. Geological Survey has evolved from informal use into a system designed to replace paper correspondence within about 5 years. A critical component in developing such a system is distributor software that accomplishes three tasks: (1) automatically verifies the recipients' addresses, (2) uses minimal computer resources, and (3) generates minimal network traffic in a distributed environment.

A distributor is an active process able to recognize certain user activities and to respond on them. In the case of electronic mail, the distributor processes messages being sent as well as those being received. Thus, the distributor allows any user to send electronic messages to any other user.

A distributor process called MAILMAN, written in FORTRAN 77 programming language, has been developed for the Division. This process, which executes in the background, is installed on all Division Prime computers, and is part of a larger system termed Electronic Documents (EDOC). When executing, the program distributes all electronic mail messages and notifies individual recipients of those messages.

Three critical steps were taken in development of the distributor. First, a data base of all electronic mail users in the Division, which automatically verifies a recipient's user identification and location, was created. In addition, group identifiers were assigned to groups of users at each site who share a common speciality and facilitate the sending of messages to thousands of recipients. Second, an algorithm was developed to transfer only one copy of a message to a site regardless of the number of recipients at that site; this maintains low network overhead. Third, the distributor was designed to use a semaphore utility, which assures that the distributor process would work only when notified by a user; this minimizes computer resource usage by the distributor process, while allowing for extremely fast response to the user.

---

[1] Hydrologist, U.S. Geological Survey, 1400 Independence Road., Rolla, MO 65401

# CHAPTER B--DIS/DISTRIBUTED ENVIRONMENT

# SUPPORTING DIFFERENT TYPES OF TERMINALS IN A DISTRIBUTED-INFORMATION ENVIRONMENT

By J. W. Atwood and S. D. Bartholoma

## ABSTRACT

The Distributed Information System (DIS) of the U.S. Geological Survey has substantially increased computing capabilities in District offices. Program implementation and data entry are no longer done by keypunching cards and developing ordered decks of interspersed instruction and data cards. Instead, programs are created and run, and data is entered interactively using video display terminals (VDT).

Programs have been written to allow data entry by "filling in the blanks" on a formatted VDT screen. Other programs use certain areas of the VDT screen to display messages. This has made computer data entry more "user friendly"; it also has caused some major problems.

Because each District office has been responsible for obtaining its own terminals, different types of terminals are used, sometimes in the same District office. In addition, some District offices also provide cooperating agencies with access to the computer; these cooperators commonly have terminals that differ from those used by the Survey. Problems result when a program written specifically for one type of terminal is run on a different type of terminal. In many cases, the program does not run on the terminal it was not written for. To compound the problem, standard cursor control subroutines do not currently (1988) exist.

With the distribution of common computer programs to all U.S. Geological Survey District offices, it is important to provide programs that are compatible with many types of terminals and to provide the ability to easily add new types of terminals as the District offices acquire them.

A cursor control subroutine library package called the DIS CURsor control package (DISCUR) has been developed, which allows any number of terminals to be supported. A set of standard subroutines has been developed, and new types of terminals can access all programs using DISCUR at the same time, and without re-compilation of the program. The use of DISCUR allows the simplest cursor control or complete control of data output and input. This report documents the development of the DISCUR control package; the report is not intended to be a user's guide.

## INTRODUCTION

In late 1982 and early 1983, the U.S. Geological Survey installed a nationwide network of about 70 multi-user minicomputers. This network, known as the Distributed Information System (DIS), has substantially increased the computing capabilities available at the District-office level. Program implementation and data entry are no longer done by keypunching cards and developing ordered decks of interspersed instruction and data cards. Instead, programs are created and run, and data is entered interactively using video display terminals (VDT).

The purpose of this report is to document the development of a cursor control package designed to assist computer programmers in writing programs that contain "user-friendly" input and output functions. This report is not intended to be a user's guide.

## DESCRIPTION OF PROBLEM

Programs have been written to allow data entry by "filling in the blanks" on a formatted VDT screen. Other programs use certain areas of the VDT screen to display messages. This has made computer data entry more "user friendly"; it also has caused some major problems.

Because each District office has been responsible for obtaining its own terminals, different types of terminals are used, sometimes in the same District office. In addition, some District offices also provide cooperating agencies with access to the computer; these cooperators commonly have terminals that differ from those used by the Survey. Problems result when a program written specifically for one type of terminal is run on a different type of terminal. In many cases, the program does not run on the terminal it was not written for; the terminal may lock up and no longer accept keyboard commands. An unreadable screen may be displayed or a somewhat readable screen may be displayed, but the cursor may not be positioned correctly to indicate the required user response. To compound the problem, standard cursor control subroutines do not currently (1988) exist.

With the distribution of common computer programs to all Geological Survey District offices, it is important to provide programs that are compatible with many types of terminals and to provide the ability to easily add new types of terminals as District offices acquire them.

## DESIRED SOLUTION

To solve the problems of using different types of terminals, a screen-control subroutine library with the following characteristics is needed:

1. The library needs to support as many types of terminals as possible.

2. Screen-control features need to be accessed by a set of standard subroutines. Programs need to be independent of the type of terminal used, with the exception of optionally requiring cursor-control and screen-control capabilities.

3. Programs using the screen-control library should not have to be re-compiled or re-loaded when a terminal is modified or a different type of terminal is added to the system.

4. The screen-control library needs to be installed as a stand-alone library at the operating-system level. It needs to be available to all programmers; it also needs to be compatible with all other programs in the computer, but not be an integral part of these programs.

5. The library needs to record the type of terminal used by a particular user, so that the user does not have to specify the terminal type every time the library is invoked.

## EXISTING PROGRAMS

Several programs for terminal control sequences have been developed and used by U.S. Geological Survey and others. Some of them are described below.

### Hard-Coded Control Sequences

Many applications in the DIS have been implemented using hard-coded, terminal control sequences. These applications, written by various programmers within the Geological Survey, support terminals using the American National Standards Institute (ANSI) terminal control sequences. The Administrative and FInancial data Management System (AFIMS) program uses this technique. Programs with hard-coded ANSI sequences can only be used on ANSI terminals such as the Digital Equipment VT100 and its equivalents (TAB, Lear-Seigler ADM36, Graphon, Tektronix, and others).

### SCREEN Program

The SCREEN program was written in the New Jersey District office of the Geological Survey in early 1980 by Stephen M. Crutchfield, a student from Drexel University. This program supported several different terminal types through the inclusion of a separate subroutine for each type of terminal. Identification of the specific type of terminal to the program was through a variable in a Fortran common block.

Adding a new type of terminal necessitated writing a new driver subroutine for that type of terminal and adding a command to invoke the new subroutine in the SCREEN subroutine. The new driver subroutine and the SCREEN subroutine then had to be re-compiled, and all programs that invoked SCREEN had to be re-loaded to make the new type of terminal available to users.

SCREEN was used in the Automatic Data Recorder (ADR) program that was installed in 1983 by the Geological Survey as an interim records-processing package. Re-named S_SCRN, it was later used in the original version of the Automatic DAta Processing System (ADAPS), which was released in 1986.

### CURCON Program

The CURCON subroutine library was written by John W. Atwood while working for the University of Utah Research Institute/Earth Science Laboratory (UURI). Programming for the first revision of CURCON was finished in January 1981. Like SCREEN, this library supported several types of terminals by including a separate subroutine for each type of terminal. The type of terminal was specified by calling a terminal-selection subroutine that identified the selected type of terminal to the program through a variable in a Fortran common block.

Like SCREEN, adding a new type of terminal meant writing a subroutine containing the necessary control sequences and placing commands to invoke the new subroutine in the initialization and driver subroutines. The CURCON subroutines then had to be re-compiled, and all of the programs using CURCON had to be re-loaded.

CURCON was used in several programs written at UURI. Later, CURCON was used in a screen entry program for creating data entry files for the Ground-Water Site Inventory (GWSI) data base on the Survey's mainframe computer in Reston, Virginia. CURCON was also used in the SOFTware EXchange (SOFTEX) program and in the GWSI and Quality of Water (QW) parts of the National Water Information System (NWIS).

### TERMINALID Program

As part of the ADAPS system, the TERMINALID program was developed by Joseph Riggsbee in the North Carolina District of the Geological Survey, with the help of Scott D. Bartholoma, Utah District office. This program maintained a data base of type of terminals keyed by user number. ADAPS applications could get a user's type of terminal without having to query the user every time the program was used. The program was later extended to include type of terminals for other, non-Survey software packages such as INFO, EMACS, and TELL-A-GRAF. This program was embedded in the NWIS software structure and was not readily available for non-NWIS applications.

## IMPLEMENTED SOLUTION

To remedy the shortcomings of the existing cursor-control programs, the DISCUR subroutine library package was written by Scott D. Bartholoma (Utah District office of the Geological Survey) and John W. Atwood (North Dakota District office). The DISCUR library provides a set of standard cursor-control functions that can be used in screen-entry and tabling programs. Support for new terminals can be added by using a definition builder to create a new terminal definition and by adding a one-line description of the terminal to a sequential file. Any program that uses the DISCUR library for cursor control will work properly on the newly added terminal without re-compilation of the program.

The DISCUR library consists of three subpackages. The first subpackage, called TERMINALID, keeps track of the user's type of terminal. It consists of a set of operating-system-level commands and a set of user-invokable subroutines to display, select, set, and retrieve terminal-type information.

The second subpackage, referred to as the C_SUBS subroutines, is used for simple cursor control only. The names of these subroutines begin with "C_", and each subroutine performs a single function. This set of subroutines may be the best choice if a user wishes to do only simple cursor control because there is no additional storage of cursor position or screen image.

The third subpackage, called CURCON, provides terminal cursor control as well as screen-entry and field-editing capabilities. The cursor position as well as a memory image of

the current screen are maintained by this group of subroutines. This set of subroutines can be used when complete control of the terminal screen is desired.

Cursor control in DISCUR is accomplished through the use of definition files. When a program using DISCUR is used, a subroutine is invoked that loads the control-code sequences into the package from the definition file of the specified terminal.

Installation of a new terminal in the DISCUR cursor-control package is relatively easy, requiring only the creation of a definition file and the addition of a line to a sequential file containing a list of available types of terminals. No re-compilation is required.

## Defining New Types of Terminals

A definition file is created with the definition-building program (BLDDEF), which creates an American Standard Code for Information Interchange (ASCII) definition file, and compiles the ASCII file into an unformatted, binary, direct-access file containing the control-code sequences to perform individual cursor movements and functions. The user's manual for the terminal usually contains a list of the control-code sequences needed for entry into the BLDDEF program.

All ASCII definition files are given the name of the terminal suffixed with ".SRC". An ASCII definition file is created by the BLDDEF program whenever the specified file name does not exist. After the ASCII definition file has been created using BLDDEF, the file can be modified using any text editor and then re-compiled by running BLDDEF again. The resulting compiled file is given the name of the terminal suffixed with ".CFG".

The subroutines C_INIT and SELTER are used to load the binary definition file into the control-code storage area. The control-code storage area is a section of data storage, internal to the DISCUR package, that contains all the control-code sequences loaded from the binary definition file. The control-code storage area provides the corresponding driver subroutines with the control-code sequences for the specified terminal. C_INIT is for use with the C_SUB group of subroutines, and SELTER is for use with the CURCON group of subroutines.

## TERMINALID Subpackage

Programs that use the DISCUR library, as well as many other programs on the Survey's minicomputers, use cursor control and require the type of terminal to be specified. It is convenient for users not to have to specify the type of terminal every time a program using cursor control is invoked. The TERMINALID subpackage provides a mechanism for the user to specify the type of terminal being used, and the information is stored for later use.

The TERMINALID package consists of two data files, three system-level commands, some utility programs, and three user-invokable subroutines. A sequential file, named DISTRICT_TERMINALS, contains an entry for each type of terminal available and, in each entry, some descriptive information and the terminal identifiers to be specified for DISCUR and

for several other proprietary programs. This is an ASCII file and may be edited to add new types of terminals and to indicate types of terminals that are available at the local computer site.

A direct-access file, named USER_TERMINALS, contains each user's terminal information filed by user number. A user on a hard-wired terminal line (as most users in the Geological Survey are) can specify the type of terminal once and change it only if another type of terminal is attached to the line. A command suitable for inclusion in a login-time command file to set the type of terminal is provided for users of a network or port-sharing device where a pool of user numbers is shared.

## System-Level Commands

Three system-level commands are associated with the TERMINALID package:

1. The TERMNL command is used to identify the type of the user's terminal for applications using the DISCUR cursor-control package and for other applications where the user's type of terminal is needed.

2. The SHOWTERM command displays the currently selected type of terminal.

3. The TERMTYPE command, which also can be invoked as a command function, extracts the type of terminal for DISCUR or any one of several other proprietary software packages, and returns it for later use.

## Utility Programs

When it becomes necessary to modify some of the information in the DISTRICT_TERMINALS file, the modifications are not automatically forwarded to users who have selected the modified type of terminal. A utility program to forward the new information to the USER_TERMINALS file, named RELOAD_TERM, has been provided.

## Subroutines

Three subroutines are provided as part of the TERMINALID subpackage that duplicate the functions of the three commands described earlier. The subroutines are:

C_SETT - Sets the type of terminal for a user;
C_TMSG - Displays the user's current type of terminal; and
C_TTYP - Retrieves terminal information.

## C_SUBS Subpackage

The C_SUBS subpackage consists of a set of subroutines used to move the cursor, erase all or part of the terminal screen, set character and line attributes, and other miscellaneous terminal-control functions. These subroutines retrieve the control sequences for the various functions from the control-code storage area that has been loaded by the initialization subroutine. The contents of the control-code storage area are loaded from a binary file that is created by the DISCUR utility "BLDDEF". If a function is not supported by the user's terminal, the subroutines return without taking any action. Following is a list of the subroutines available in the C_SUBS subpackage, grouped by function.

Initialization is controlled by the subroutine C_INIT. It retrieves the DISCUR driver name from the TERMINALID package and loads cursor-control information. The programmer may specify that only terminals that support cursor control may be used. If the selected terminal does not support cursor control, an appropriate code is returned. This subroutine needs to be used before any other C_SUBS cursor-control subroutines are used.

These subroutines control cursor movement:

C_UP - Moves cursor up;
C_DOWN - Moves cursor down;
C_LEFT - Moves cursor left;
C_RGHT - Moves cursor right;
C_POSN - Moves cursor to a specified location; and
C_HOME - Moves cursor to "home" position (top left corner).

These subroutines control screen erasure:

C_EBOL - Clears from beginning of line;
C_EEOL - Clears to end of line;
C_ELIN - Clears entire line;
C_EBOS - Clears from beginning of screen;
C_EEOS - Clears to end of screen;
C_EALL - Clears entire screen, cursor is not moved; and
C_CLS - Clears entire screen and moves cursor to "home" position.

These subroutines control screen attributes:

C_DB - Bold;
C_DBF - Bold, flashing;
C_DBFR - Bold, flashing, reversed;
C_DBR - Bold, reversed;
C_DBU - Bold, underlined;
C_DBUF - Bold, underlined, flashing;
C_DBUR - Bold, underlined, reversed;
C_DU - Underlined;
C_DUF - Underlined, flashing;
C_DUFR - Underlined, flashing, reversed;
C_DUR - Underlined, reversed;
C_DF - Flashing;

C_DFR - Flashing, reversed;
C_DR - Reversed;
C_DALL - Bold, underlined, flashing, reversed; and
C_DOFF - All attributes off, normal characters.

These subroutines control line attributes:
C_LNDT - Double width, double height, top one-half;
C_LNDB - Double width, double height, bottom one-half;
C_LNDW - Double width, single height; and
C_LNOF - Attributes off - Single width, single height.

These subroutines are used for miscellaneous functions:
C_LOCK - Lock lines;
C_SCRL - Set scrolling region;
C_NARO - Set to narrow screen;
C_WIDE - Set to wide screen;
C_AXON - Auxiliary (printer) port on;
C_AXOF - Auxiliary (printer) port off;
C_BELL - Ring terminal bell; and
C_RSET - Perform master reset.

## CURCON Subpackage

CURCON is the primary subroutine within the CURCON subpackage. CURCON is used to perform all of the cursor-positioning functions. Other subroutines are used to: load the control-code storage area with the proper cursor-control sequence codes (SELTER); select an option (LISOPT); enter data (REDSTR, REDTMP, REDNUM, REDDAT, REDTIM, REDSCR); display an error message (MESQUR); and access individual values within the control-code storage area (RTRVPR, STROPR).

For terminals that support only cursor positioning, CURCON can emulate most of the other clearing and scrolling functions. Emulation of the clearing and scrolling functions requires strict control of the cursor position. This is why the cursor position and screen image are stored when using CURCON. The mixing of other screen control subroutines with CURCON, although possible, is not recommended because they modify the screen image and move the cursor without updating the stored screen image and cursor position variables. Fortran "read" and "write" statements used to perform terminal input and output should also be avoided when using CURCON, for the same reason. All output to the screen needs to be through the CURCON, MESQUR, and LISOPT subroutines. All entry from the terminal needs to be through one or more of the REDxxx subroutines. Otherwise, the program will not work properly on terminals that require the emulation of certain functions.

The subroutine SELTER needs to be used to initialize the control-code storage area before any of the other CURCON subroutines are used. SELTER has no arguments. The type of the user's terminal is read from a system file that first needs to be set using the TERMNL command at the operating-system level. If the type of the user's terminal is undefined, SELTER will

display a list of the available type of terminals and prompt the user to select one. The list of terminals given is dependent both on the DISTRICT_TERMINALS file and on whether the user is a local or remote user. If the user is on a hard-wired line, only the terminals that have been defined for the local office are listed. If the user is a remote user, the entire list of available terminals is displayed.

CURCON is used to perform all cursor positioning and attribute setting. Areas within the screen can be blanked, filled, or edited. A scrolling region can be set in both the vertical and horizontal directions. Messages are displayed in normal, bold, underlined, reverse video, and double-sized characters if the terminal supports such attributes.

LISOPT is used to display an option list and accept a decision on the desired option.

MESQUR is used to display an optional message at the bottom of the screen and pause for permission to continue. If the user's program detects an error, a message can be displayed using MESQUR. The message is displayed on the last line, and a prompt to continue is displayed on the line above the message. Special messages and prompts can be displayed in the same manner. MESQUR also displays a two-line command description at the bottom of the screen. The description lines are displayed on the same two lines used to display messages, and replace the messages after exiting the MESQUR subroutine. If desired, the command description lines can be displayed initially or re-stored after a screen clear by using a special command to MESQUR. The command description lines contain a visual reminder of a few of the field-editing commands available to the user. These field-editing commands are used by the data-entry subroutines, as well as LISOPT and MESQUR, to perform within-field editing.

The data-entry subroutines REDSTR, REDTMP, REDNUM, REDDAT, REDTIM, and REDSCR are used for entering data of various types. All data needs to be entered using these subroutines so that the program can always maintain a record of the cursor location. With the exception of REDSCR, the data-entry subroutines allow within-field editing. REDSCR is a special subroutine that reads a field of specified length directly from the stored screen image. All entered data is in character-string form. If the entered data is a numerical value, it should be read first by REDNUM, and then an internal read (from a character variable, instead of from a file) should be performed on the character string to retrieve the required numeric value. The field-editing codes that define field positioning and edit modes are passed to the invoked subroutine for proper execution. This allows the invoked subroutine to position to the proper field, change to edit or entry mode, store the data, or abort.

The last subroutines, RTRVPR and STORPR, are used to access the individual values in the DISCUR control-code storage area. Care needs to be taken in using these subroutines, as erroneous modification of the DISCUR system variables can have unexpected results. These subroutines are provided so that programmers can access the values contained in the control-code storage area. These subroutines give direct access to the control sequences as well as the stored screen.

## CONCLUSIONS

With the increasing use of VDT terminals for entry of data to a computer, there is a need for the use of standard terminal-control subroutines. The need for supporting many types of terminals is great. The DISCUR control package has been developed to fit these needs. DISCUR provides a complete set of terminal control features and allows easy addition of any number of types of terminals. Addition of new terminals to the system is independent of any program using the DISCUR package. To use a newly added type of terminal, the user need only select the new terminal before running programs containing commands to DISCUR. DISCUR is a stand-alone package that can be used in any program that performs cursor control.

# CHAPTER C--GIS APPLICATIONS IN USGS

# Displaying Data from the National Water Data Exchange by Use of a Geographical Information System

Bruce Parks
U.S. Geological Survey
Office of Water Data Coordination
Reston, Virginia

## ABSTRACT

The capability exists to plot the location of the water-data-collection sites for each State by using a geographic information system program with appropriate background files. Information from the Master Water Data Index of the National Water Data Exchange, a data base maintained at the U.S. Geological Survey's headquarters office in Reston, Virginia, has been transferred into a series of digital files in a geographical information system that represents features on maps. "Key files" and special commands allow rapid selection of records by type of data collected, collecting agency, frequency of data collection, or any combination of selection criteria. The relation and structure of the files and practical applications in which they could be used are examined.

## BACKGROUND

The Office of Management and Budget Circular A-67 gives the Department of the Interior the responsibility to maintain a catalog of water-data information on a national basis. The National Water Data Exchange (NAWDEX) is an interagency program residing in the U.S. Geological Survey to facilitate the exchange of water data and to promote the standardization of procedures for handling water data. Federal, State, local government organizations, academic institutions, and private organizations that collect and use water data are participants in the NAWDEX program. NAWDEX maintains a Master Water Data Index (MWDI), which is a computerized index of available water data. This index resides on an Amdahl mainframe computer at the Geological Survey Headquarters office at Reston, Virginia.

Requests for information from NAWDEX may be made through the 76 NAWDEX Assistance Centers located throughout the country. Responses to requests are usually given in tabular form, although limited capability exists for showing the areal distribution of sites. For the novice user of NAWDEX, retrievals can be time consuming and may require resubmittal to get the exact information or format required. The purpose of this paper is to describe a scheme for transferring data from NAWDEX to a geographical information system that can be used interactively to display the information on maps, and to identify some applications for using the system.

Water-resources managers and data users need to be able to readily see the data that are available for their area of interest. This is best accomplished by using a computerized information management system that allows the display of information on maps and modification of the information being displayed. ARC/INFO, a Geographical Information System (GIS) developed by the Environmental Systems Research Institute of Redlands, California, is a management tool that can be used to organize, manipulate, and graphically display the geographic data available in the MWDI.

## APPROACH

To display the location of sites listed in the MWDI, data files need to be developed that contain the information in the MWDI and reside in the data base structure of a GIS. Software has been developed that will read all information in the MWDI for a set of sites and write the information to tape. The tape is then loaded onto a Prime minicomputer where a program edits and formats the file, builds a data base, and creates an ARC/INFO coverage, which is a digital analog of a single map sheet stored in a suite of files. INFO is a relational data base software program developed by HENCO Software, Inc., Waltham, Massachusetts, which is used to build the data base. ARCPLOT, a subsystem of ARC/INFO, is then used to display the location of the sites or subsets of sites along with other geographical data showing State and county boundaries, hydrologic units, river locations, and other information, that has been drawn from other data files. The subsets can be controlled by selecting a range of values for any of the parameters or items listed for the sites.

### Structure of the Master Water Data Index

The MWDI data base on a mainframe computer is maintained through a data base management system called SYSTEM 2000. The data in the SYSTEM 2000 data base are organized into a hierarchical structure as shown in figure 1. The MWDI contains the following general categories of information:

Station Identifiers

- Unique identifiers (site number, agency number, NAWDEX number)
- Operating organization
- Geographical identifiers (country, State, county, hydrologic unit)
- Type of site (stream, well, lake, spring, estuary)
- Physical identifiers (drainage area, basin descriptor)
- Station status (active, inactive)
- Supplementary data available

Type of water data collected

- Surface water
- Ground water
- Water quality
- Meteorological

Attributes of water-data collected

- Period of record
- Record continuity
- Data parameters collected
- Frequency of data collection
- Data storage media
- Purpose of activity
- Status of activity

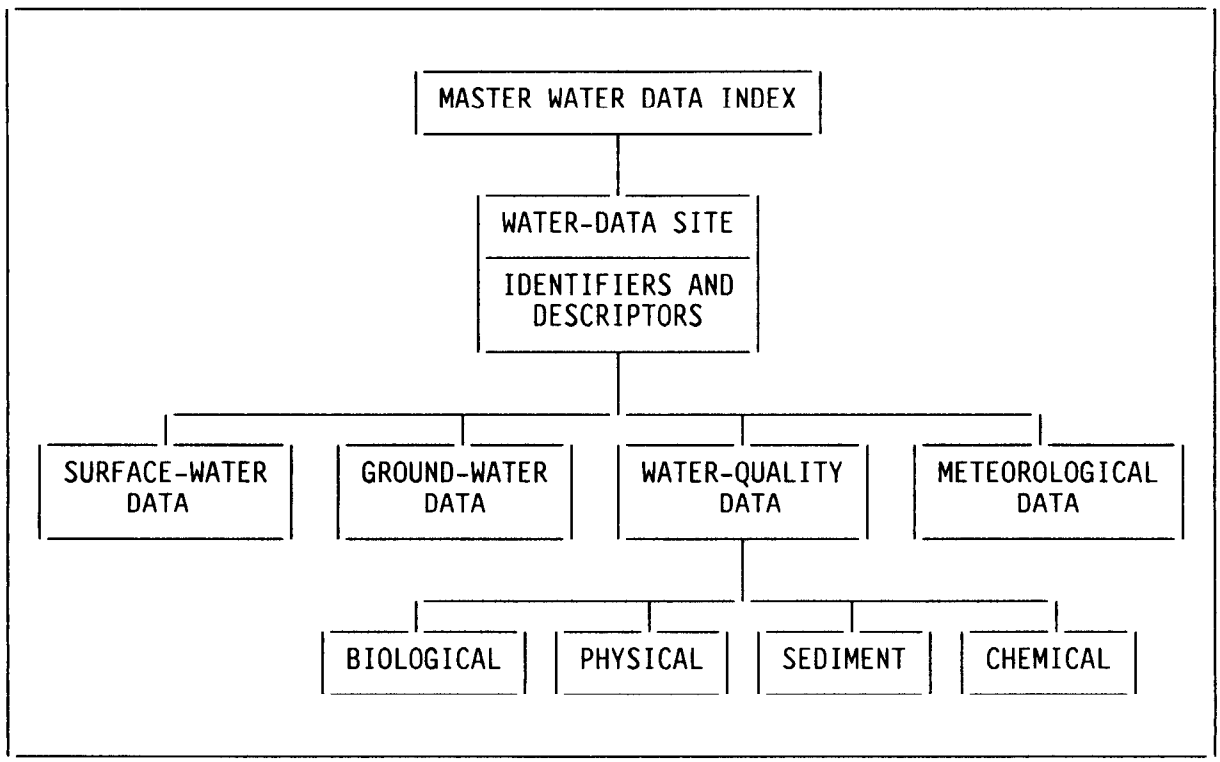


Figure C1. Structure of the Master Water Data Index's SYSTEM 2000 data base.

Transfer of Data from the National Water Data Exchange to the Prime

The MWDI resides on the Amdahl computer at the Geological Survey in Reston, Virginia. Job Control Language (JCL), a computer software program, was used to make retrievals by State or territory for all available data from each site and to write the information to tape. Prime's magnetic tape utility program, MAGNET, was used to read the file into the Prime environment and the GET COPY command in INFO was used to transfer the data into an INFO file. An ARC/INFO coverage was created by using the GENERATE command in ARC/INFO with the latitudes and longitudes from the MWDI data, and by projecting the coverage into Albers equal-area coordinates.

The MWDI contains more than 200 parameters for each of the 460,000 sites listed for the United States. The data were categorized by State to keep the files in a more convenient size. To maintain continuity between the original data base and the data base to be created for the GIS environment, a parallel system was developed in ARC/INFO. A directory, or suite of files, was created for each State. The component names used in the SYSTEM 2000 data base have been used as item names in the ARC/INFO data base and the groups of components used in the SYSTEM 2000 data base have been organized as separate files in the ARC/INFO data base. The suite of ARC/INFO files that correspond to the structure shown in figure C1 are given in table C1. Directories that contain two sub-directories -- named INFO and LOC -- were established for each State. The INFO directory contains the normal ARC/INFO files plus the separate data files for the types of data available. The LOC directory is the ARC/INFO coverage containing the files that define the attributes or boundaries of points, lines, and polygons in the coverages. Both directories are accessed through the ARC/INFO software.

### Application

To display the data, ARCPLOT is used with the new command KEYSELECT, described by Lanfear and Hitt (1988). Key files and lookup tables can be created by using common items such as SW_KEY, GW_KEY, and so forth, found in the LOC.PAT file and KEY-LOC which is found in the other files (table 1). Using the searching techniques described by Lanfear and Hitt (1988), the user can select a group of records based on availability, non-availability, or magnitude of any of the parameters in any of the INFO files in a relatively short period of time.

There are a number of coverages in Geological Survey files that can be used as base maps to display the states, counties, rivers, water bodies, and hydrologic units. The sites selected from the NAWDEX files can be overlayed on these maps to show the spatial distribution of the sites. Because there are codes in the NAWDEX files that list the State, county, hydrologic unit, and names for the sites, discrepancies in the data base can be detected by overlaying sites listed for a certain area on a polygon outlining that area, whether by State, county, hydrologic unit or combination.

## SUMMARY

An index of water-data information for the United States currently resides in the U.S. Geological Survey's National Water Data Exchange and is maintained on an Amdahl computer at the Geological Survey's headquarters in Reston, Virginia. To have the information readily available for use in a GIS environment, the data were transferred to a Prime computer, copied into INFO files, and developed into ARC/INFO coverages for use with the ARCPLOT subroutines. The files were set up so that keyfiles and lookup tables could be used to decrease the time required for selecting and reselecting data. Because the individual files can be fairly large, the KEYSELECT command is used to decrease the time for selecting the desired group of records.

TABLE C1. List of items in each file in INFO directory

| FILE NAME: LOC.PAT | FILE NAME: SW_DATA | FILE NAME: GW_DATA |
|---|---|---|
| AREA | NAWDEX# | NAWDEX# |
| PERIMETER | SW_BEGIN_YR | GW_BEGIN_YR |
| LOC# | SW_END_YR | GW_END_YR |
| LOC-ID | SW_INTERRUPTED | GW_INTERRUPTED |
| NAWDEX# | SW_OWDC_NO | GW_OWDC_NO |
| NAWDEX_ID | SW_OWDC_SEQ | PRIN_AQUIFER |
| NAWDEX_AGCY | COMPLETE | AQUIFER_TYPE |
| AGCY_STA_NO | PEAK_STAGE | LEVEL_FREQ |
| STATION_NAME | LOW_STAGE | LEVEL_MED |
| NON_US_COUNTRY | STAGE_MED | DISCHRG_FREQ |
| STATE | COMPLETE_FLOW | DISCHRG_MED |
| COUNTY | PEAK_FLOW | SUBSIDE_FREQ |
| HYDROL_UNIT | LOW_FLOW | SUBSIDE_MED |
| CONG_DIST | MISC_FLOW_MEAS | DEPTH_OF_WELL |
| SITE_TYPE | FLOW_MED | GW_RECMD_MTHDS |
| BASIN_DESCRP | VOLUME | GW_OTHER |
| WDSD_OFC_CODE | VOLUME_CHANGE | MAJOR_VAR |
| DRAINAGE.AREA | VOLUME_MED | GW_TELEMETRY |
| NC_AREA | SW_UNIT_FLOW | GW_LST_UPDATE |
| LAST_UPDATE | SW_UNIT_STAGE | GW_PURPOSE |
| STATE_COUNTY | SW_UNIT_VOLUME | GW_RECORDER_TYPE |
| PRIMARY_USE | SW_RECMD_MTHDS | GW_RECORDER_FREQ |
| WRD_ACCT | SW_OTHER | GW_PN_CODE |
| DOWNSTREAM_ORDER | SW_TELEMETRY | GW_MODIFIERS |
| OTHER_DATA | SW_LST_UPDATE | KEY-LOC ** |
| SW_ACTIVE | SW_PURPOSE | |
| SW_KEY * | SW_RECORDER_TYPE | |
| GW_ACTIVE | SW_RECORDER_FREQ | |
| GW_KEY | SW_PN_CODE | |
| QW_ACTIVE | SW_MODIFIERS | |
| QW_KEY | KEY-LOC ** | |
| BIO_ACTIVE | | |
| BIO_KEY | | |
| PHY_ACTIVE | | |
| PHY_KEY | | |
| SED_ACTIVE | | |
| SED_KEY | | |
| CHM_ACTIVE | | |
| CHM_KEY | | |
| MET_ACTIVE | | |
| MET_KEY | | |
| MISC_INFO_KEY | | |

* Physical record number of corresponding record in the SW_DATA file, for use with INFO's RELATE...LINK command. Other KEYs are similar. Not all stations in the LOC.PAT file have corresponding records in the QW_DATA, GW_DATA, or similar files; KEYs for these records are set to ZERO.

** Physical record number of corresponding record in the LOC.PAT file.

TABLE C1. List of items in each file in INFO directory -- Continued

| FILE NAME: QW_DATA | FILE NAME: CHM_DATA | FILE NAME: BIO_DATA |
|---|---|---|
| NAWDEX# | NAWDEX# | NAWDEX# |
| QW_BEGIN_YR | SOLIDS_DIS | ENTERIC_BACT |
| QW_END_YR | MAJOR_IONS | NATIVE_BACT |
| QW_INTERRUPTED | HARDNESS | PHYTOPLANKTON |
| QW_OWDC_NO | SILICA | ZOOPLANKTON |
| QW_OWDC_SEC | PHOSPHORUS | PERIPHYTON |
| QW_RECMD_MTHDS | PHOS_SPECIES | MACROPHYTON |
| QW_TELEMETRY | NITROGEN | MICROINVERTS |
| QW_LST_UPDATE | N_SPECIES | MACROINVERTS |
| QW_PURPOSE | DETERGENTS | VERTEBRATES |
| QW_RECORDER_TYPE | OMI_CONSTITS | FUNGI |
| QW_RECORDER_FREQ | RADIOACTIVITY | VIRUSES |
| QW_PN_CODE | RCHM_SPECIES | BIO_RECMD_MTHDS |
| STORET_POINTER | CARBON | BIO_BEGIN_YR |
| QW_MODIFIERS | ORG_GROUPS | BIO_END_YR |
| KEY-LOC ** | PEST_SPECIES | BIO_LST_UPDATE |
| | OTH_ORG_SPECIES | BIOLOGIC_MED |
| | BIOCHEM_OX_DMND | PRIMARY_PRDCTVTY |
| | CHEM_OX_DMND | SCENDARY_PRDCTV |
| | DISSOLVED_OX | CHEMOSYNTHETIC_A |
| | OTHER_DIS_GAS | BIOSTIMULATORY_T |
| | CHEM_RECMD_MTHDS | TOXICITY_TEST |
| | CHM_BEGIN_YR | OTHER_BIOASSAY_T |
| | CHM_END_YR | CHM_TISSUE_ANALY |
| | CHM_LST_UPDATE | HISTOPATH_ANALYS |
| | CHEMICAL_MED | OTHER_TISSUE_ANA |
| | CHM_MODIFIERS | BIO_MODIFIERS |
| | KEY-LOC ** | KEY-LOC ** |

** Physical record number of corresponding record in LOC.PAT file.

TABLE C1. List of items in each file in INFO directory -- Continued

| FILE NAME: PHY_DATA | FILE NAME: SED_DATA | FILE NAME: MET_DATA |
|---|---|---|
| NAWDEX# | NAWDEX# | NAWDEX# |
| TEMPERATURE | BED_LOAD | MET_BEGIN_YR |
| SPEC_CONDUCT | CNCNTRIN_SUS | MET_END_YR |
| TURBIDITY | CNCNTRIN_TOT | MET_INTERRUPTED |
| COLOR | PART_SIZ_SUS | MET_RAINFALL |
| ODOR | PART_SIZ_BED | MET_UNIT_RAINFAL |
| PH | SED_DIS_SUS | MET_AIR_TEMPERAT |
| SUSPD_SOLIDS | SED_DIS_TOT | MET_RSVD1 |
| PHY_RECMD_MTHDS | SED_RECMD_MTHDS | MET_WIND_VELOCIT |
| PHY_BEGIN_YR | SED_BEGIN_YR | MET_RSVD2 |
| PHY_END_YR | SED_END_YR | MET_RSVD3 |
| PHY_LST_UPDATE | SED_LST_UPDATE | MET_RECMD_MTHDS |
| PHYSICAL_MED | SEDIMENT_MED | MET_OTHER |
| PHY_MODIFIER | SED_MODIFIERS | MET_TELEMETRY |
| KEY-LOC ** | KEY-LOC ** | MET_LST_UPDATE |
| | | MET_MEDIA |
| | | MET_RECORDER_TYP |
| | | MET_RECORDER_FRE |
| | | MET_PN_CODE |
| | | MET_MODIFIERS |
| | | KEY-LOC ** |

** Physical record number of corresponding record in LOC.PAT file.

REFERENCES

Lanfear, Kenneth J., and Hitt, Kerie J., 1988, Efficient operations on large geographic information system coverages: in Proceedings of the Eighth Annual ESRI User Conference, March 21-25, 1988, Palm Springs, California, [Redlands, California] Environmental Systems Research Institute, 12 p.

# USING A GEOGRAPHIC INFORMATION SYSTEM TO DETERMINE PHYSICAL BASIN CHARACTERISTICS FOR USE IN FLOOD-FREQUENCY EQUATIONS

By

James J. Majure and P.J. Soenksen

## ABSTRACT

*A set of computer programs, named Basinsoft, was developed to use digital cartographic data to compute basin characteristics that are hypothesized to be related to floods. The programs work in conjunction with a proprietary geographic information system. Three digital cartographic data sets (coverages) of drainage basin boundaries, streams, and selected topographic contours are used by Basinsoft to calculate 16 basin characteristics: total drainage area, noncontributing drainage area, contributing drainage area, main stream length, basin length, channel slope, basin slope, basin width, shape factor, total stream length, drainage density, basin relief, ruggedness number, basin perimeter, relative relief, and drainage frequency. Basinsoft also plots the three coverages along with selected basin characteristics. The results produced by Basinsoft may be directly useful to hydrographic and geomorphologic studies, but its ultimate value will be to provide basin characteristics that can be used to improve flood-discharge and flood-frequency predictions.*

## INTRODUCTION

Determining the magnitude and frequency of floods at any site is an important step in the economical planning and safe design of bridges, culverts, levees, and retention structures, and is essential for the management of flood plains. At gaging stations where flood data have been collected for a number of years, these determinations can be easy to make. However, the great majority of basins in Iowa have no flood data, and methods that transfer data from other sites need to be used.

Previous investigations by the U.S. Geological Survey on the magnitude and frequency of floods in Iowa were prepared by Schwob (1953, 1966) and Lara (1973, 1987). The methods illustrated in these reports used data that were regionalized according to general basin types that were assumed to produce similar floods. Equations relating one or more measurable basin characteristic (drainage area, channel slope, mean annual precipitation) to the various flood-frequency characteristics were determined from available flood data within each region. Flood characteristics at ungaged sites were then estimated by using the appropriate regional equation and measured basin characteristics at the ungaged site.

The physical characteristics of a basin (size, ruggedness, drainage pattern, shape, soils, and so forth) largely control flood characteristics in that basin. However, limited cartographic data and the tedious procedures required to compute many physical characteristics of a basin, have prevented researchers from

using all but a few of these characteristics in equations that estimate flood frequency. The composite effects of the remaining characteristics were dealt with in a general way through the use of large geographic regions with generally similar characteristics. However, there are significant limitations to this approach. Because the defined regions are not entirely homogeneous, and physical characteristics do not change abruptly at regional borders, individual basins, especially small basins, can be of a different type than that of the region they are in. In such cases the normal regional equations are not applicable, and there is no quantitative way of determining which, if any, of the other regional equations are applicable.

A more direct approach is to incorporate several basin characteristics into the equations so that flood characteristics can be directly related to quantitative measurements. Implicit to this approach is the capability to compute a variety of physical basin characteristics for a large number of basins where flood data have been collected, so the equations can be developed, and flood characteristics are desired. The advent of digitized cartographic data and geographic information systems (GIS) makes such large-scale computations not only possible, but also fast and accurate. This report presents the results of an investigation that used aGIS to determine physical basin characteristics for use in flood-frequency equations.

## DETERMINING PHYSICAL BASIN CHARACTERISTICS USING A GEOGRAPHIC INFORMATION SYSTEM

### Concepts

A set of computer programs, named Basinsoft, was developed to use digital cartographic data to compute basin characteristics that are hypothesized to be related to floods. This is accomplished by creating a representation of a drainage basin on a computer using the ARC/INFO1/geographic information system. Once the ARC/INFO representation of the drainage basin is entered into the computer, the information maintained by ARC/INFO and the computational capabilities it provides make it possible to calculate many basin characteristics.

The first step in creating a representation of a drainage basin is to define those characteristics of a basin that are to be determined. Because the number of basin characteristics that affect floods is large, it is not practical to determine all characteristics. For Basinsoft, the most important characteristics were chosen. These characteristics require that the contributing and noncontributing drainage areas, the streams, and the topography of a basin be represented. After determining which aspects of a basin are to be represented, how to best represent these aspects in the GIS needs to be determined. Geographic features are represented in ARC/INFO as polygons, lines, or points. A complete basin in Basinsoft is represented by three ARC/INFO coverages: a polygon coverage for the drainage area, a line coverage for the streams, and a line coverage for the topography.

The concept of having a complete basin in Basinsoft is important because when a command is given and the specified basin is not complete, Basinsoft will display an error message and abort. Specifically, a complete basin named <basin>, would consist of: a drainage area coverage, <basin>.BAS; a stream coverage, <basin>.STR; and a topography coverage, <basin>.CON. A complete basin also includes an INFO file named

<basin>.CHAR, that stores the basin characteristics calculated by the commands. If any of these four components of a Basinsoft basin are missing, the basin is not considered complete.

## The User Interface

The Basinsoft commands are a set of ARC macros that work with short INFO programs to calculate basin characteristics from basin representations. The macros reside in a subdirectory of the ARC/INFO workspace in which the basin coverages are maintained. The INFO programs and the characteristics file reside in the workspace INFO directory.

To enter the system, execute the BASINSOFT.CPL file in the workspace. This CPL enters ARC/INFO, establishing the Basinsoft macros as system commands. It accepts, as an option, any legal command in ARC. A station file for the ARC/INFO session also may be specified after -STATION, -STAT or -ST. A station file identifies the hardware to be used by ARC/INFO. For example, the command: **OK, CPL BASINSOFT ARCPLOT -STATION IOWA**
would enter ARC/INFO, establish the Basinsoft macros as system commands, set the hardware characteristics to those specified in the IOWA station file, and enter the ARCPLOT subsystem.

The basic command format for the Basinsoft commands is:
<char> <basin> {recno} where:
<char> is the abbreviation of the characteristic to be calculated,
<basin> is the name of the basin, and
{recno} optionally specifies the record number in the characteristics file where the calculation will be stored.
For example, the command:
**ARC: BS WILL 3**
will calculate the basin slope (BS) for the basin named WILL, and will place the results of the calculation in the third record of the characteristics file. If a record number is specified, it needs to exist in the characteristics file.The {recno} option is included for flexibility. It allows multiple sets of characteristics to be maintained for a single basin. If omitted, the last record in the file will be updated. The contents of the updated record are overwritten.

Many of the characteristics calculated by Basinsoft use data that are automatically maintained by ARC/INFO, such as the area of polygons and the length of arcs. It is assumed that the coverage units are feet. The commands apply appropriate conversions to calculate the desired units.

The complete set of Basinsoft commands includes:

**BN** - basin name; a descriptive1- to 40-character basin name. Entered after prompt.

**TDA** - total drainage area, in square miles, including noncontributing areas.

**NCDA** - drainage area, in square miles, that does not contribute to surface runoff at the basin outlet.

**CDA** - drainage area, in square miles, that contributes to surface runoff at the basin outlet.

**BS** - average basin slope, in feet per mile,

which is computed as: BS = (length of all contours in miles) x (contour interval in feet) / CDA.

**SL** - stream length, in miles, is the length of the main stream from basin outlet to end of the defined channel.

**BL** - basin length, in miles, SL plus the extension of the defined channel to the basin divide.

**CS** - main channel slope, in feet per mile, is the difference in elevation at points 10- and 85-percent of BL divided by the distance between those points.

**BW** - effective basin width, in miles, which is computed as BW = CDA/BL.

**SF** - shape factor, dimensionless, is the ratio of effective basin width to basin length, which is computed as SF = BW/BL.

**TSL** - total length of streams, in miles, which is computed by summing the lengths of all stream segments within the CDA.

**DD** - drainage density, in miles per square mile, which is computed as DD = TSL/CDA within the CDA.

**BR** - basin relief, in feet, is the difference between the highest and lowest points within CDA. The user is prompted for these elevation values.

**RN** - ruggedness number, in feet per mile, which is computed as RN = TSL x BR/CDA.

**BP** - basin perimeter, in miles, is the length of the entire basin divide.

**RR** - relative relief, in feet per mile, which is computed as RR = BR/BP.

**DF** - drainage frequency, in stream segments (arcs) per square mile, computed as DF = (number of stream segments)/CDA within CDA.

**BASINC** - calculates the entire set of basin characteristics for a basin.

**MAPBASIN** - creates a map of the basin and the following ten characteristics: TDA, CDA, BS, CS, SF, DD, BR, RN, RR, and DF. An example of MAPBASIN output is shown in figure 1.

**ADDCHREC** - adds a record to the characteristics file.

**STARTBASIN** - is a utility program to help set up a basin that is to be digitized from one 7.5-minute quadrangle. It prompts for the projection of the quadrangle and the smallest latitude and longitude values. It then creates empty basin coverages, with coverage units feet, and the characteristics file.

## Coverage And File Structure

As explained above, each basin in Basinsoft is represented by three ARC/INFO coverages. The three coverages need to be maintained in feet and named as follows: <basin>.BAS, <basin>.STR, and <basin>.CON. The characteristic file needs to be named <basin>.CHAR. In each case, <basin> is a 1-to 6-character basin name. For example, the basin named WILL consists of the three coverages, WILL.BAS, WILL.STR, and WILL.CON, and the characteristic file named WILL.CHAR.

**Figure 1.--An example of MAPBASIN output.**

Williams Cr nr Petersville, IA

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| TDA | = | 1.795 | $mi^2$ | DD | = | 2.18 | $mi/mi^2$ |
| CDA | = | 1.795 | $mi^2$ | BR | = | 145 | ft |
| BS | = | 380.40 | ft/mi | RN | = | 315.61 | ft/mi |
| CS | = | 0.00 | ft/mi | RR | = | 26.17 | ft/mi |
| SF | = | 0.593 | | DF | = | 5.01 | streams/$mi^2$ |

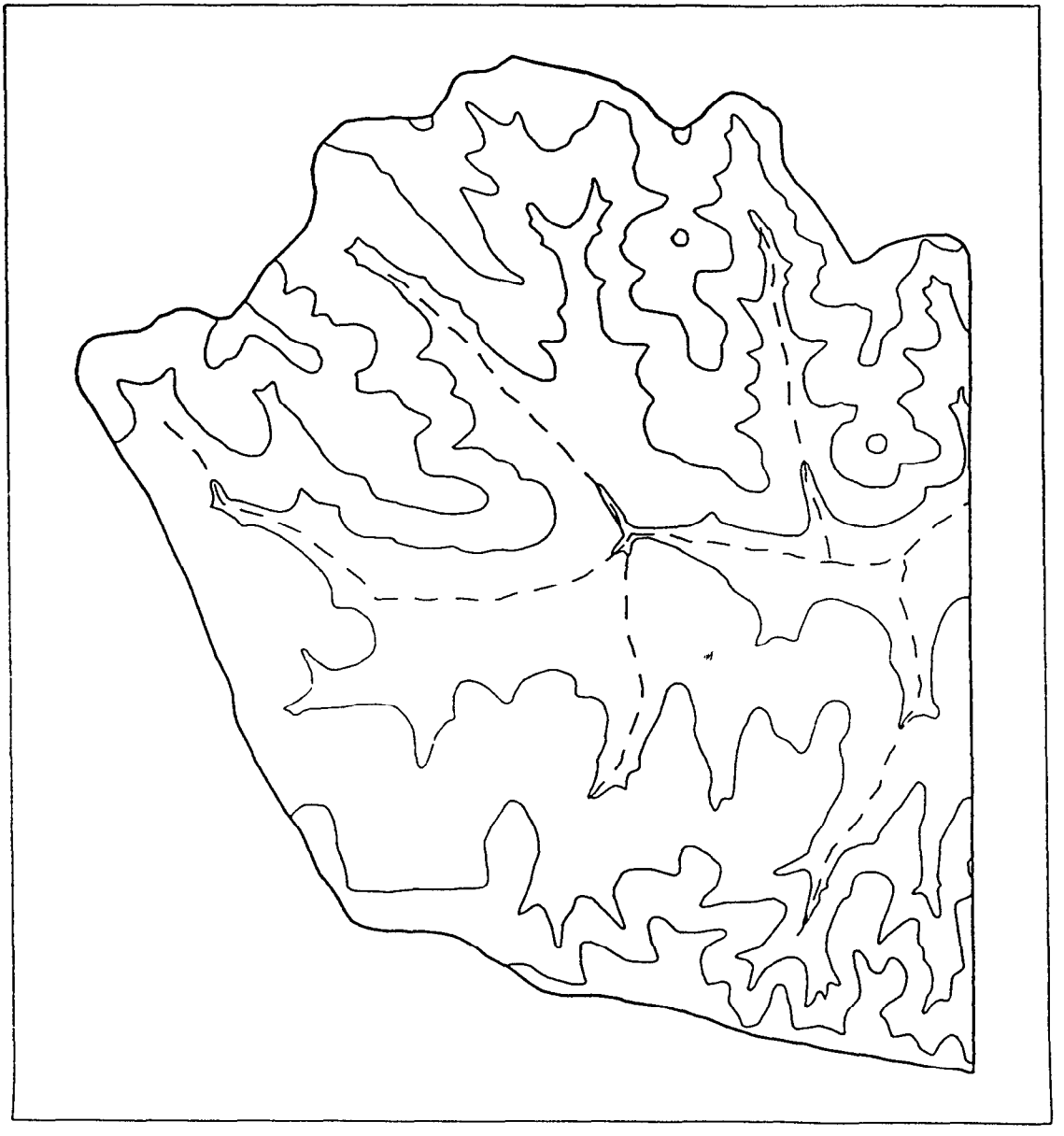

Figure 2.--File structure of the <basin>.BAS.PAT file

DATAFILE NAME: <basin>.BAS.PAT
5 ITEMS: STARTING IN POSITION 1

| COL | ITEM NAME | WDTH | OPUT | TYP | N.DEC | ALTERNATE NAME |
|---|---|---|---|---|---|---|
| 1 | AREA | 4 | 12 | F | 3 | |
| 5 | PERIMETER | 4 | 12 | F | 3 | |
| 9 | <basin>.BAS# | 4 | 5 | B | - | |
| 13 | <basin>.BAS-ID | 4 | 5 | B | - | |
| 17 | CONTRIB | 1 | 1 | I | - | |

The <basin>.BAS coverage is a polygon coverage that defines the basin area. For most basins, this coverage will contain one arc that defines the perimeter of the basin. For those basins that have noncontributing drainage areas, this coverage will contain additional polygons identifying the noncontributing areas. The item, CONTRIB, in the coverage PAT file will be used to determine whether a polygon represents a contributing or noncontributing drainage area. A value of 0 in CONTRIB indicates a noncontributing drainage area and a value of 1 indicates a contributing drainage area. The structure of the <basin>.BAS.PAT file is shown in figure 2.

The <basin>.STR coverage is a line coverage of all streams in the basin. The <basin>.STR.AAT file has the item CODE in addition to those maintained by ARC/INFO. This attribute can contain one of three values: 0, secondary stream; 1, main stream; and 2, extension line. All of the arcs that comprise the one main stream have a CODE value of 1. All other arcs that represent streams have a CODE value of 0. There is only one arc with a CODE value of 2. This arc is added to the end of the main stream and extends to the basin divide. This arc does not represent a stream, but is used to calculate the BL characteristic. The structure of the

Figure 3.--Structure of the <basin>.STR.AAT file

DATAFILE NAME: <basin>.STR.AAT
8 ITEMS:STARTING IN POSITION 1

| COL | ITEM NAME | WDTH | OPUT | TYP | N.DEC | ALTERNATE NAME |
|---|---|---|---|---|---|---|
| 1 | FNODE# | 4 | 5 | B | - | |
| 5 | TNODE# | 4 | 5 | B | - | |
| 9 | LPOLY# | 4 | 5 | B | - | |
| 13 | RPOLY# | 4 | 5 | B | - | |
| 17 | LENGTH | 4 | 12 | F | 3 | |
| 21 | <basin>.STR# | 4 | 5 | B | - | |
| 25 | <basin>.STR-ID | 4 | 5 | B | - | |
| 29 | CODE | 1 | 1 | I | - | |

Figure 4.--Initial structure of the <basin>.CON.AAT file.

DATAFILE NAME: <basin>.CON.AAT
8 ITEMS: STARTING IN POSITION 1

| COL | ITEM NAME | WDTH | OPUT | TYP | N.DEC | ALTERNATE NAME |
|---|---|---|---|---|---|---|
| 1 | FNODE# | 4 | 5 | B | - | |
| 5 | TNODE# | 4 | 5 | B | - | |
| 9 | LPOLY# | 4 | 5 | B | - | |
| 13 | RPOLY# | 4 | 5 | B | - | |
| 17 | LENGTH | 4 | 12 | F | 3 | |
| 21 | <basin>.CON# | 4 | 5 | B | - | |
| 25 | <basin>.CON-ID | 4 | 5 | B | - | |
| 29 | ELEV | 4 | 4 | I | - | |

<basin>.STR.AAT file is shown in figure 3.

The <basin>.CON coverage is a line coverage that contains a series of topographic contours for the basin. There can be as many contours as desired as long as the interval is constant. Initially, the <basin>.CON.AAT file has the item, ELEV, in addition to those maintained by ARC/INFO. ELEV will contain the elevation of each contour arc. The initial file structure of the <basin>.CON.AAT file is shown in figure 4.

Once the <basin>.BAS coverage and the <basin>.CON coverage have been created, coded, and cleaned up, the ARC/INFO IDENTITY command needs to be run on <basin>.CON, using <basin>.BAS as the identity coverage. The resulting coverage will contain all the attributes of both initial coverages. All items in the <basin>.CON.AAT file, except those maintained by ARC/INFO, ELEV, and CONTRIB must be deleted and the coverage renamed to <basin>.CON.This effectively allows contours that plot within

Figure 5.-- The final structure of the <basin>.CON.AAT file

DATAFILE NAME: <basin>.CON.AAT
9 ITEMS : STARTING IN POSITION 1

| COL | ITEM NAME | WDTH | OPUT | TYP | N.DEC | ALTERNATE NAME |
|---|---|---|---|---|---|---|
| 1 | FNODE# | 4 | 5 | B | - | |
| 5 | TNODE# | 4 | 5 | B | - | |
| 9 | LPOLY# | 4 | 5 | B | - | |
| 13 | RPOLY# | 4 | 5 | B | - | |
| 17 | LENGTH | 4 | 12 | F | 3 | |
| 21 | <basin>.CON# | 4 | 5 | B | - | |
| 25 | <basin>.CON-ID | 4 | 5 | B | - | |
| 29 | ELEV | 4 | 4 | I | - | |
| 33 | CONTRIB | 1 | 1 | I | - | |

noncontributing areas to be differentiated from those that do not. The final structure of the <basin>.CON.AAT file is shown in figure 5.

## Arc Macro And INFO Program Structure

One of the most attractive features of this set of programs is its simplicity. Most of the information needed to calculate the basin characteristics is automatically maintained by ARC/INFO. Often, all that is needed are some simple calculations and unit conversions. Each command to calculate a basin characteristic consists of an ARC macro and a short INFO program. Almost all of the ARC macros are identical except that they initiate different INFO programs. The INFO programs are identical except for several lines that calculate the specific basin characteristic. Adding new basin characteristics can be done by adding an appropriate item to the characteristics file, and modifying an existing ARC macro and INFO program to calculate the new characteristic.

The ARC macro, TDA.AML, is shown in figure 6, and the INFO program, TDA.PG, is shown in figure 7. Comments are included indicating which lines are changed to create other commands.

Note that due to limitations in INFO one slightly unorthodox programming technique was needed in order to keep the programs generic or able to operate on a user specified basin and not a specific one.In these programs, INFO data files are SELECTed using the following two lines:
**CONCATENATE $CHR2 FROM 'SEL ',$CHR1,'rest of file name' EXEC $CHR2**
The SELECT command is created from the basin name stored in $CHR1, and is put in $CHR2. The command in $CHR2 is then executed by the EXEC command. Because the value of the variable executed by the EXEC command is not determined until run time, no file is explicitly selected during compilation and subsequent data item references will cause errors. To resolve this problem, duplicate INFO data files have been set up with the same file definitions as the corresponding actual files and are SELECTed prior to the EXEC command. Because a file with the proper structure has been SELECTed, compilation finishes without errors. A consequence of this method of selecting files is that anytime the file definition of an INFO file referenced by one of the programs changes, the definition of the corresponding duplicate file also needs to reflect the change and all programs that reference the file need to be recompiled. Otherwise, errors will occur, but no warning will be given!

Figure 6.--The ARC macro, TDA.AML

```
&args basin recno
&s basin [translate %basin%]
&if [null %basin%] &then
        &return &warning Usage:TDA
        <basin> {recno} /* This line
        changes...
&if ^ ( [exists %basin%.bas -coverage] &
        [exists %basin%.con -coverage] ~
        & [exists %basin%.str -coverage]
        & [exists %basin%.char -info] )
        &then &return &warning Basin,
        %basin%, is incomplete or does not
        exist.
&if [null %recno%] &then &s recno 0
&if ^ ( [type %recno%] = -1 ) &then
        &return &warning "recno" must be
        an integer record number.
&system como -ntty
down info
&data info
ARC
```

```
RUN TDA.PG /* and this line changes
%basin%^
%recno%
Q STOP
&end
up
&system como -tty
&return
```

Figure 7. --The INFO program, TDA.PG.

```
PROGRAM TDA.PG
FO $CHR1,8,8,C
FO $CHR2,80,80,C
FO $NUM21,4,12,F,3
FO $NUM23,4,4,I
ACC $CHR1     /* Receive the basin name
ACC $NUM23  /* Receive the char file
      recno
CMD COMO -TTY
SEL FAKE.CHAR ;use following EXEC to
      select real file;
CONCATENATE $CHR2 FROM 'SEL',
      $CHR1,'.CHAR'
EXEC $CHR2
IF $NOSEL EQ 0
      ADD TDA FROM ADD.DUMMY
ENDIF
      IF $NUM23 NE 0
      RES BY $RECNO = $NUM23
      IF $NOSEL EQ 0
         CONCATENATE $CHR2 FROM
         'Specified record doesn't exist in ',
         $CHR1,'.CHAR.'
         DIS $CHR2
         GOTO END
      ENDIF
ENDIF
SEL FAKE.BAS.PAT ;use following EXEC
      to select real file;   /*These
CONCATENATE $CHR2 FROM 'SEL
       '.$CHR1,'.BAS.PAT'/*lines
EXEC $CHR2              /*change
RES BY $RECNO  = 1    /*for each
CALC $NUM21 = 1 - AREA
                        /*characteristic.
SEL FAKE.CHAR ;use following EXEC to
      select real file;
CONCATENATE $CHR2 FROM 'SEL
       ',$CHR1,'.CHAR'
EXEC $CHR2
IF $NUM23 EQ 0
      RES BY $RECNO EQ $NOSEL
ELSE
      RES BY $RECNO EQ $NUM23
ENDIF
CALC TDA = $NUM21    .00000003587
      /* This line also changes.
LABEL END
MO -NTTY
END
```

## Efforts To Automate Data Entry

Once the coverages representing a basin are in place, the Basinsoft commands are easy and straightforward. Digitizing the basins, however, can be a tedious and lengthly process.In order to decrease the time spent on this process, several possible ways of automating the data entry have been investigated.One possibility is acquiring digital line-graph data for hydrography and topography. This method still requires someone to define the drainage-basin perimeter and any noncontributing drainage area in the basin. Another possibility is acquiring digital elevation-model data. Programs exist that can determine the basin perimeter as well as any noncontributing drainage area, generate streams, and generate topographic contours from digital elevation-model data. This information can then be converted into ARC/INFO format and entered into the Basinsoft programs. The biggest problem with both of these possibilities is the major task of producing this digital data for all of the areas that need to be analyzed.

## CONCLUSIONS

Basinsoft offers a means of relating flood-frequency predictions more directly to physical characteristics of individual drainage basins. Basinsoft provides a framework to which additional data describing the physical characteristics of a basin can be added as such data are obtained. This should ultimately help make predictions of flood frequency as accurate as possible.

## REFERENCES

Lara, O.G., 1973, Floods in Iowa--Technical manual for estimating their magnitude and frequency: Iowa Natural Resources Council Bulletin 11, 56 p.

------1987, Method for estimating the magnitude and frequency of floods at ungaged sites on unregulated rural streams in Iowa: U.S. Geological Survey Water-Resources Investigations Report 87-4132, 34 p.

Schwob, N.H., 1953, Iowa floods, magnitude and frequency: Iowa Highway Research Board Bulletin I, 171 p.

------1966, Magnitude and frequency of Iowa Floods: Iowa Highway Research Board Bulletin 28, 45 p.

# CHAPTER D--COMPUTER GENERATED GRAPHICS

# THE INTEGRATION OF COMPUTER GRAPHICS AND TEXT-EDITING PROGRAMS

By Donald R. Block[1]

## ABSTRACT

It is not possible to edit text parts of a graphics display while in CA-TELLAGRAF because it does not have an internal text editor. A FORTRAN-77 program called NCTELAGRAF was written to enable the user to retain graphics screen display during a CA-TELLAGRAF session while internally using PRIMOS command-file text editors without leaving CA-TELLAGRAF. This program allows use of text editors EM, ED, and WM and the RUN and SYS commands within CA-TELLAGRAF. The RUN command sends the edited file to CA-TELLAGRAF for execution. The SYS command invokes the PRIMOS emulator, which permits the user to enter commands directly to the operating system.

## INTRODUCTION

Computer-generated graphics is rapidly becoming a standard method of producing graphs and charts. In the U.S. Geological Survey, computer-generated graphs and charts are used to display and help interpret hydrologic data. Computer software packages that enable the analyst to easily produce such graphs and charts are required tools in many hydrologic investigations. Many of the computer-generated graphics produced by Survey personnel are products of Computer Associates' CA-TELLAGRAF or CA-DISSPLA software run on PRIME minicomputers. CA-TELLAGRAF is primarily used for producing graphs from small data sets and for presentation graphics. CA-DISSPLA is a group of FORTRAN subroutines used by programmers to perform repetitive tasks requiring interaction of graphs with large data bases. CA-TELLAGRAF is relatively user friendly when compared to CA-DISSPLA.

CA-TELLAGRAF can be run in an interactive or batch mode. During an interactive session, CA-TELLAGRAF receives commands directly from the user, whereas during a batch session, it reads commands from a previously created command file. The most common methods of creating command files are: (1) using the "SAVE." command during an interactive CA-TELLAGRAF session, (2) entering CA-TELLAGRAF commands into a file using a text editor, and (3) running other software packages that create command files, such as Computer Associates' CueChart.

Advantages of using command files to run CA-TELLAGRAF are that (1) they can be easily modified to make changes in a graph and (2) they are often data-independent, allowing the production of the same style graph for different sets of data without modification. In the Survey's North Carolina, Nevada, and Tennessee District offices, libraries of command files are used by District personnel to produce graphs (R.C. Massingill, U.S. Geological Survey, written commun., 1988; J.C. Stone, U.S. Geological

---

[1]Computer Programmer Analyist, U.S. Geological Survey, 3916 Sunset Ridge Road, Raleigh, North Carolina 27607; telephone: (919) 571-4000.

Survey, oral commun., 1988). The disadvantage of using command files is that the exact command file required for a particular hydrologic application commonly is not available in the library, and modifying one of these command files using CA-TELLAGRAF is time consuming. Because CA-TELLAGRAF has no internal text editor, modifications are performed in an interactive manner. The user makes changes to a command file using an available external editor and then runs CA-TELLAGRAF to see the results of these changes. If further modifications are necessary, the user must repeatedly "quit" CA-TELLAGRAF, make the modifications, and then reinvoke CA-TELLAGRAF.

The time consumed alternating between CA-TELLAGRAF and external editors can frustrate the user. To alleviate this frustration, a FORTRAN-77 program called NCTELAGRAF[2] has been written to allow the use of Marc Software's WordMarc Composer+ and PRIME computer's EMACS and ED editors during an interactive CA-TELLAGRAF session. This paper describes the operation and programming of NCTELAGRAF.

## NCTELAGRAF USER INFORMATION

NCTELAGRAF is invoked by typing "NCTELAGRAF filename". The filename argument refers to the user's CA-TELLAGRAF command file. If the filename argument is omitted, the program prompts the user to enter the name of his command file. Once the filename is entered, that file becomes the selected file inside NCTELAGRAF.

Once invoked, NCTELAGRAF submits the selected file to CA-TELLAGRAF for execution. After the selected file has been executed, control is returned to NCTELAGRAF. NCTELAGRAF looks and operates like CA-TELLAGRAF. Unlike CA-TELLAGRAF, NCTELAGRAF does not automatically clear graphics from the terminal screen after viewing. This allows the user to refer to the graphic while editing the command file. The user may clear the graphics area at any time using the terminal's graphics erase function. NCTELAGRAF also automatically appends a period to all user-entered commands before submitting them to CA-TELLAGRAF for execution, with the exception of the five NCTELAGRAF commands: EM, ED, WM, RUN, and SYS.

The EM, ED, and WM commands invoke the text editors made available by NCTELAGRAF. The EM command invokes the EMACS full screen editor. The ED command invokes the PRIMOS line editor. The WM command invokes the WordMarc Composer+ full screen editor. The RUN command submits the selected file to CA-TELLAGRAF for execution. The EM, ED, WM, and RUN commands can be entered with or without a filename argument. If the commands are entered with no filename argument, the selected file becomes the object of the command's action. If the filename argument is entered with the command, then that file becomes the selected file and is the object of the command's action.

The SYS command invokes NCTELAGRAF's "PRIMOS emulator". While in PRIMOS emulation, NCTELAGRAF imitates PRIMOS by allowing the user to submit commands directly to the operating system. To quit PRIMOS emulation and return to NCTELAGRAF, the user must type "Q". The user can also quit a NCTELAGRAF session and return to PRIMOS by typing "Q".

---

[2]All references to NCTELAGRAF refer to NCTELAGRAF version 2.0.

## NCTELAGRAF PROGRAMMER INFORMATION

NCTELAGRAF is a Fortran-77 program that invokes CA-TELLAGRAF as a subroutine using the PRIME subroutine CP$. NCTELAGRAF submits all commands except EM, ED, WM, RUN, and SYS to CA-TELLAGRAF using a COMI (command input) file for an input buffer. This method has been described by Block and Gordon (1988, p. 100).

NCTELAGRAF submits command files to CA-TELLAGRAF as control files with the command "CONTROL FILE filename." where the filename argument is the pathname of the user's selected CA-TELLAGRAF command file. CA-TELLAGRAF is exited by default when a control file completes execution. To prevent this, NCTELAGRAF appends the line "RESET. CONTROL FILE KB." to all command files before submitting them to CA-TELLAGRAF for execution. Immediately after the control file has been executed by CA-TELLAGRAF, NCTELAGRAF takes the terminal out of graphics mode to prevent CA-TELLAGRAF from clearing the graphics area after the graphic is drawn. Prior to allowing user input, the command "CONTROL FILE 'DUMMY'." is submitted to CA-TELLAGRAF for execution. This command opens a control file named "DUMMY", which contains no information, but opening it causes CA-TELLAGRAF to close the user's selected command file. The selected command file can then be opened for editing by NCTELAGRAF.

The NCTELAGRAF commands EM, ED, WM, and SYS use the PRIME subroutine CP$ to submit commands to the PRIMOS operating system. When the process invoked by one of these commands completes execution, control is returned to NCTELAGRAF. The NCTELAGRAF RUN command checks the selected file for the line "RESET. CONTROL FILE KB.", appends this line if necessary, and then submits the commands "RESET. RESET. CONTROL FILE filename." to CA-TELLAGRAF. These commands instruct CA-TELLAGRAF to reset all variables to their default values before executing the control file. As before, after the control file has been executed, the command "CONTROL FILE 'DUMMY'." is submitted to CA-TELLAGRAF and results in the closing of the user's selected command file.

When the user types a "Q" in NCTELAGRAF, the command is submitted to CA-TELLAGRAF, ending the CA-TELLAGRAF session. This causes the CP$ subroutine that invoked CA-TELLAGRAF to return control to the main section of NCTELAGRAF allowing it to complete execution.

## NCTELAGRAF SYSTEM INFORMATION

The NCTELAGRAF.F77 program is delivered with the accompanying files:

NCTELAGRAF.READ.ME.1ST - the installation instructions and user's manual,

NCTELAGRAF.INSTALL.CPL - a program to compile and bind NCTELAGRAF.F77, and

NCTELAGRAF.USERS - a list of people to whom the software has been sent.

The files occupy 14 PRIME records of disk space. The software can be installed for a single user or for multiple users. NCTELAGRAF requires the following conditions to operate correctly without modification:

1. The commands EM, ED, and WM invoke PRIME's EMACS, PRIME's ED, and Marc Software's WordMarc Composer+, respectively.

2. The file named TAGPRO.DAT exists in the directory from which NCTELAGRAF is invoked, and this file is correct for the terminal being used. Refer to the CA-TELLAGRAF user's manual for more instructions on setting up a TAGPRO.DAT file.

3. Every user has an active abbreviations file. This is required for NCTELAGRAF's PRIMOS emulator because NCTELAGRAF inserts "AB -EE" at the beginning of all commands submitted to PRIMOS. This allows the PRIMOS emulator to expand user abbreviations and will cause an error if no active abbreviations file exists.

NCTELAGRAF works with Revision 6.0 of CA-TELLAGRAF at Revision 2.0 of PRIMOS and is available upon request to the District Chief, U.S. Geological Survey, 3916 Sunset Ridge Road, Raleigh, North Carolina 27607. Installation instructions are included.

## SUMMARY

Many computer-generated graphics produced by Survey personnel make use of CA-TELLAGRAF software for producing graphs and other presentation graphics. However, because CA-TELLAGRAF has no internal text editor, the editing of text within the graphics is a time-consuming iterative process whereby the user must exit CA-TELLAGRAF each time to make text modifications and then reenter the program to view the results of changes.

The NCTELAGRAF program essentially provides an internal text-editing capability during operation of the CA-TELLAGRAF program without exiting from CA-TELLAGRAF. Thus, on-screen text changes may be made and results immediately evaluated. The EM, ED, and WM commands are the text editors made available by NCTELAGRAF. Operating commands RUN and SYS, respectively, submit the edited file to CA-TELLAGRAF for execution and invokes NCTELAGRAF's PRIMOS emulator. NCTELAGRAF-user, programmer, and system details are provided.

## REFERENCES

Block, D.R., and Gordon, J.A., 1988, COLLAGE: a menu-driven graphics editing software package, *in* Proceedings of computer associates graphics users' group meeting, Nashville, Tenn., Feb. 29-Mar. 1, 1988: p. 93-102.

Integrated Software Systems Corporation, 1985, TELL-A-GRAF user's manual, version 5.0: San Diego, Calif., Integrated Software Systems Corporation.

# USE OF THE GRAPHICAL KERNEL SYSTEM STANDARD

# FOR HYDROLOGIC APPLICATIONS

By Thomas C. Wood and Alan M. Lumb

## ABSTRACT

The Graphical Kernel System is an internationally accepted standard for computer graphics programming that allows for maximum portability of applications between different hardware platforms. Although the Graphical Kernel System standard is language-independent, it provides specific language bindings, such as Fortran, which are used for source code development. The Graphical Kernel System standard addresses the main functional areas of computer graphics programming, including output, coordinate systems, segmentation, interactive input, and "meta" file generation. Graphical Kernel System implementations are being developed on an increasing number of hardware platforms. The implementations most useful to the U.S. Geological Survey are those on Prime and UNIX-based workstations.

A set of Fortran utilities have been developed using the Graphical Kernel System for a range of surface-water programs used by the Geological Survey. Graphics include time plots at scales of minutes to years, x-y plots, and probability plots. Non-time axes can be arithmetic or logarithmic. Probability plots with Gaussian transformations can be fraction, percent, or recurrence interval. The utilities are stored in a Geological Survey software library available on most Prime systems. Implementations also have been made on a UNIX-based workstation and a personal computer.

# INTRODUCTION

The Graphical Kernel System (GKS) is an internationally accepted standard for computer graphics programming that allows for maximum portability of applications between different hardware platforms. GKS was developed as a result of an increasing demand for computer graphics standards that began during the late 1960's, when new technology made computer graphics available to more people than ever before.

Graphics displays and plotters were developed and in use by the early 1950's. Color displays were in use by 1962. Most conventional graphics hardware had emerged by 1965. However, few people were involved with computer graphics. It is estimated that only 100 display systems were installed worldwide in 1965, at an average cost of $400,000 each (Hopgood and others, 1983, p. 2). Hardware vendors provided software that took advantage of their particular hardware capabilities. This led to very different approaches to writing graphics software.

## Growing Demand for Standards

During the late 1960's, new technology produced interactive graphics devices at much lower costs, which made graphics available to a larger population. These new devices offered new capabilities, yet there was still no standard approach to writing software to address them. The lack of a common approach to developing graphics systems led to the formation of different schools of graphics systems design. The different systems were generally (1) device-dependent, (2) application-dependent, and (3) environment-dependent (Enderle and others, 1987, p. 54).

During the early 1970's, the cost of graphics devices continued to plummet and their use spread rapidly. Graphics systems developers were growing tired of re-writing their software with each hardware change. Also, people wanted to exchange programs with their colleagues without re-writing every routine at the local installation. Computer graphics seemed ready for an attempt at standardization.

## Standardization Activities

During the middle 1970's, numerous standards organizations around the world began developing ideas for graphics standards. Major efforts were undertaken in Germany, the United States, and at the international level. Portability of application programs was the primary target of the standardization attempts. This would be achieved by standardizing the interface between the application program and a combined set of graphics functions. Three main strategies emerged for developing this standard interface (Enderle and others, 1987, p. 54):

(1) Design a new graphics language. It is very difficult to introduce a new programming language.

(2) Design a graphics extension of an existing high-level language. This is easier, however, the compiler would require modification to add new

features and overhead. It is difficult to persuade users of a language to accept new overhead.

(3) Design a subroutine package, callable from existing high-level languages. Defining graphics functions using parameter lists is not very appealing, however, this is the only strategy that could be implemented without harming other interests. Therefore, a subroutine package became the vehicle for implementation of the GKS and other proposed graphics standards.

In Germany, the Committee for Information Processing founded the subcommittee "Computer Graphics" in 1975. This 13 member group actually developed the initial GKS. The first draft of GKS was published in 1977.

In the United States, the Association for Computing Machinery's Special Interest Group on Graphics (ACM-SIGGRAPH) founded its own Graphics Standards Planning Committee in 1975. The Graphics Standards Planning Committee developed another subroutine package, known as the CORE system.

On the basis of increasing interest in graphics standards around the world, the International Organization for Standards formed a graphics committee in 1977, named WG2, with the goal of producing an international standard. In 1978, WG2 reviewed both the CORE and GKS proposals. In 1979, GKS was chosen as the starting point for an international standard. The International Organization for Standards managed the long GKS review and approval process.

### Graphical Kernel System Becomes an International and American Standard

In August 1985, the International Organization for Standards published GKS as an international standard. Soon after, the American National Standards Institute adopted GKS as an American National Standard. In November 1986, American National Standard GKS was adopted as a Federal Information Processing Standard, for use in the Federal Government.

## DEFINITION OF THE GRAPHICAL KERNEL SYSTEM

GKS is a hardware-independent, language-independent, device-independent, application interface for two-dimensional graphics output and interactive graphics input, providing maximum portability of applications. GKS contains all basic functions for interactive and non-interactive graphics on a wide range of graphics equipment.

### Hardware Independence

GKS is implemented through a subroutine package, which is called from a high-level language, such as Fortran. Practically all hardware platforms support at least one high-level language. Therefore, GKS is hardware independent.

### Language Independence

The GKS standard defines its capabilities through abstract subroutine names, such as "POLYLINE", "SET TEXT HEIGHT", and so forth. GKS implementors translate these abstract subroutine names into high-level language specific

subroutine names, such as "GPL", "GSCHH", and so forth, in Fortran. GKS may be implemented in any high-level language that provides the subroutine construct. Therefore, GKS is language independent.

## Device Independence

The GKS standard is at such a level of abstraction that device peculiarities are shielded from the application program(American National Standard for Information Systems, 1985, p. 12). An important concept that helps GKS be device independent is the logical workstation. The GKS interface provides uniform input and output functions that are defined on logical workstations. There exists an interface that assigns the abstract input and output functions defined on the logical workstations to specific device drivers.

## Main Components of the Graphical Kernel System

GKS addresses the main functional areas of computer graphics programming. It specifies standard output primitives that are used to generate pictures. Three coordinate systems are provided, as well as transformations between each. GKS also provides segmentation and interactive input, which when combined together, add a new dimension to computer graphics systems. The logical workstation concept allows for device-independent graphics programming. Two metafile formats are supported for picture transfer.

### Output Primitives

One of the basic functions of a graphics system is to generate pictures. In GKS, pictures are composed of output primitives (fig. D6). The main output primitives are:

(1) Polyline: a line connecting a list of specified positions,

(2) Polymarker: a symbol drawn at a specified position,

(3) Text: a character string drawn at a specified position,

(4) Fill Area: a polygon filled with a specified color or pattern.

Primitive attributes, such as polyline color, text height, and so forth, may be specified individually by the application program through attribute setting routines. GKS also provides attribute bundle indices and tables for each primitive type as an alternative. Each output device available to GKS has a set of primitive bundle tables associated with it. The application specifies a bundle index number for each primitive type and leaves the attribute selection to the bundle table (fig. D7). An example might be drawing a line on a graph. The application specifies "POLYLINE BUNDLE INDEX 2", followed by "DRAW POLYLINE." If a color device is specified as the workstation, the line is drawn in red on the basis of the attributes specified in the bundle table for that device. If a monochrome device is specified as the workstation, the line is drawn in a dashed pattern in white.

## Coordinate Systems and Transformations

GKS defines three coordinate systems that are used to display and manipulate graphics on any device (fig. D8). The first of these is world coordinates, a cartesian coordinate system. The application defines world coordinates by specifying units of choice in the X and Y direction. GKS defines an intermediate coordinate system, normalized device coordinates with abstract values of 0 to 1 in both X and Y directions. The third coordinate system used by GKS is the device-dependent device coordinates, which GKS determines from the device being used. A normalization transformation is used to map world coordinate space into normalized device coordinate space. Normalized device coordinate space is then mapped to device coordinates space by a workstation transformation.

An example might be an application that produces a graph with units of 0 to 100 on both X and Y axes, which the programmer determines by the data being portrayed. The programmer defines world coordinates as 0 to 100 in both X and Y directions. The output device for the program might be any number of graphic devices with different sizes of output area. In order for the graph to be output correctly on any device available to GKS, the specified world coordinates need to be transformed to device coordinates, which varies depending on the device. This is done through the intermediate coordinate system, normalized device coordinates. First, the world coordinates ranges of 0 to 100 are transformed into an arbitrary normalized device coordinate space, which ranges from 0 to 1 in both X and Y directions. Then GKS can transform the arbitrary normalized device coordinates space into device coordinates of the specified device and the graph is drawn correctly.

## Segmentation

As stated earlier, GKS pictures are composed of output primitives, such as lines, text, and so forth. Output primitives may be grouped together in parts and may be addressed and manipulated by the application as single units called segments. An example of creating a segment with GKS abstract routines is:

```
CREATE SEGMENT (1)
DRAW POLYLINE (5,X,Y)
DRAW TEXT (XPOS,YPOS,'SEGMENT 1')
CLOSE SEGMENT.
```

Segments may be manipulated by changing their transformation. GKS provides utilities that can be used by the application to move, scale, and rotate segments, through segment transformations. Segment attributes can also be changed by the application. These attributes include:

(1) Visibility: indicates whether or not a segment is being displayed on the display surface.

(2) Highlighting: a segment may be emphasized in a device-independent way by changing its appearance. An example is blinking or emboldening the primitives that make up the segment.

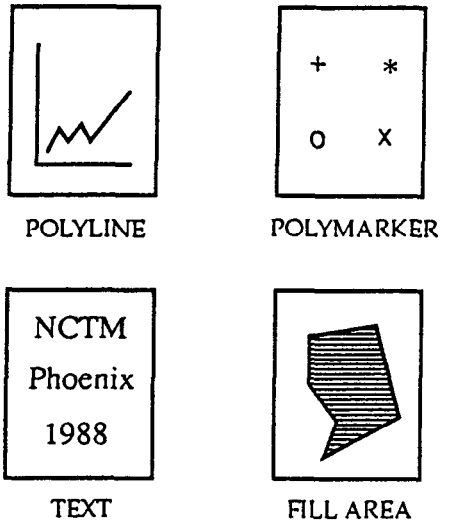


Figure D6.--Graphical Kernel System standard output primitives.

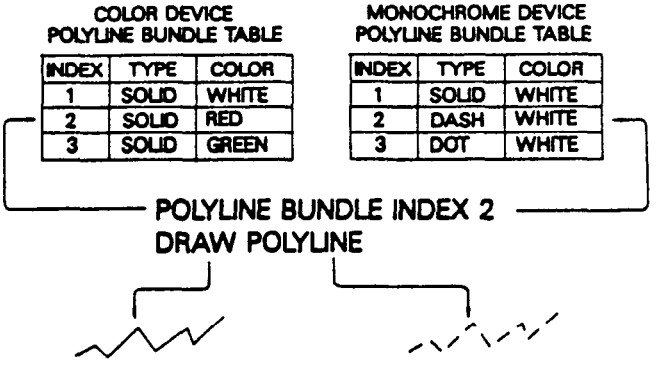


Figure D7.--Device independent output. Specifying "POLYLINE BUNDLE INDEX 2" produces a solid red line on color devices and a dashed line on monochrome devices, based on attributes set up in bundle table.

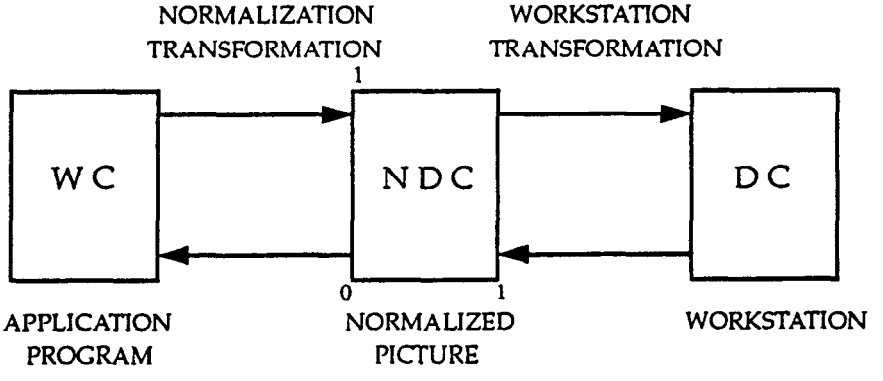


Figure D8.--Graphical Kernel System coordinate systems and transformations.

(3) Detectability: indicates whether or not a segment can be selected by the pick input function.

Segment manipulation works well on terminals that support segmentation in their hardware, such as Tektronix 4107 and higher models. GKS provides software emulation of segmentation on devices lacking segmentation, such as Tektronix 4010. However, segment manipulations performed through software are considerably slower than those managed by hardware.

## Interactive Input

GKS provides a method for the application to request user input. This interactive input, combined with segment manipulation that is based on the input, adds a new dimension to computer graphics systems. A user of such a system is able to point to a certain part of the display (segment), move, erase, scale, rotate, and so forth, the selected part of the display.

GKS allows user input from several different logical input device classes, which are abstractions of physical devices; this provides device-independent input capability. Logical input devices are mapped to physical input devices, on the basis of a device driver selected by the application. Logical input classes include:

(1) Locator: provides a position in world coordinates, supplied by the user positioning a locator device. An example of a locator input device is a crosshair cursor moved around on a display by thumbwheels.

(2) Pick: provides a segment name, determined by the user positioning a locator on an output primitive contained in the segment.

(3) Valuator: provides a real number, supplied by the user, to be used for scaling, rotation, and other segment manipulation. Entering a number into a keyboard is the most common valuator input device.

(4) Choice: provides an integer which corresponds to the user's selection from a number of choices. These choices might be assigned to function keys on a keyboard, for example.

## Logical Workstation Concept

GKS provides an application with device independence through the logical workstation concept. The logical workstation is an abstraction of the physical device. The application contains uniform input and output definitions that are mapped from the logical workstation to the actual device being used. A small section of program code defining the physical devices available to the application may be set up separately from the body of the application. The body of the application contains abstract references to input and output and is, therefore, completely portable. Input is referenced by logical input devices. Output is referenced by general drawing primitives and primitive attribute bundle tables (figs. D7 and D8).

GKS implementations provide some form of a workstation reference document, which details device-specific information about each device available to the particular GKS implementation. This information includes

the bundle tables for each primitive type, input devices available for each logical input class, whether segmentation is available through hardware, and so forth.

### Metafiles

Graphics metafiles provide a method of storing and retrieving pictures in an external file in a device and application-independent manner. GKS provides an interface to two metafile formats, GKS Metafile and Computer Graphics Metafile. Both metafile formats are available to GKS applications by specifying a corresponding workstation identifier. These identifiers are listed in the workstation reference document.

GKS Metafile is a format used only by GKS and is not part of the international standard, but is heavily used by GKS applications. A GKS Metafile records all output, including segment information, in a sequential format. GKS applications are able to read a GKS Metafile and display all or part of the stored picture, completely under user control.

The Computer Graphics Metafile is an American National Standards Institute standard and a draft international standard currently being processed by the International Organization for Standards. The Computer Graphics Metafile is being implemented by a rapidly increasing number of software vendors as an optional output format for graphics systems. A Computer Graphics Metafile stores a static picture and does not include segment information.

## IMPLEMENTATION OF THE GRAPHICAL KERNEL SYSTEM

Practical implementation of GKS is done through language bindings that have been developed for several programming languages. There is a GKS implementation on most hardware platforms.

### Language Bindings

GKS is defined in a language-independent manner, specifying abstract routines for graphical functions. These abstract routines are translated into language-specific routines for each particular programming language. This language-dependent layer is called a language binding. GKS has been translated into Fortran 77, PASCAL, C, and ADA programming languages. Each of these language bindings are also standardized in order to preserve portability of the applications. Standard routine names are derived from abbreviations for each corresponding GKS function. In the Fortran language binding, all routines begin with the letter "G" and end with up to five letters as an abbreviation for the function of the routine. For example, the GKS input function "REQUEST LOCATOR" is translated to routine "GRQLC".

### Hardware Platforms

GKS has been implemented by numerous hardware and software vendors on most platforms, from microcomputers to supercomputers. Purchase prices for GKS implementations range from $500 to greater than $30,000. GKS

implementations of most interest to the U.S. Geological Survey are those available for Prime systems and UNIX-based workstations.

The Geological Survey currently has a GKS implementation on every Prime in the Distributed Information System (DIS) network. Computer Associates' CA-GKS, using the Fortran binding, was introduced as part of the DISSPLA revision 10.0 upgrade in 1985. CA-GKS, although a separate package from DISSPLA, does share files with DISSPLA, including device drivers and metafiles. Other companies that offer GKS packages on the Prime include NOVA*GKS by Nova Graphics International, D-PICT/GKS by Pansophic Systems, and GRAFPAK-GKS by Advanced Technology Center.

The graphics requirements of DIS-II, the Geological Survey's next generation of computing tools, includes a GKS implementation. The main component of DIS-II will be a UNIX-based workstation. There is a rapidly increasing number of vendors offering GKS implementations on UNIX-based workstations. These include all of the packages mentioned above for Prime systems, as well as GK-2000 by Precision Visuals, Inc., and Visual: GKS by Visual Engineering.

## WHEN TO USE THE GRAPHICAL KERNEL SYSTEM

When designing a graphics application, the existence of one or more of the following situations warrants consideration of using GKS for the application development (Federal Information Processing Standard, 1986, p. 2):

(1) The application will be programmed centrally for a decentralized system that might consist of different types of computers and graphics devices.

(2) It is probable that the life of the application will be longer than the life of the present graphics equipment.

(3) The application is likely to be used by organizations outside the Federal Government, such as state and local governments, universities, and so forth.

(4) The application will be run on equipment other than that on which it is developed.

(5) The application will be maintained by programmers other than the original ones.

### Advantages of Developing an Application Using a Graphical Kernel System Implementation

(1) Maximum portability of applications between different hardware platforms.

(2) Limited dependence on a specific software vendor. GKS implementations may be purchased from numerous vendors.

(3) Access to new devices with minimal program modification, due to the logical workstation concept.

(4) Program development in commonly used languages, such as Fortran.

### Disadvantages of Developing an Application Using a Graphical Kernel System Implementation

(1) More programming required to produce output than in most proprietary graphics packages. For example, drawing a chart axis requires individual calls to primitive drawing routines to draw the axis line, each tick mark, and the tick labels. A GKS program might require ten or more lines of code to draw an axis. A package like DISSPLA requires only one line of code to produce the entire axis.

(2) Additional code is required to translate logical workstation commands to those of a particular device. This additional overhead will sometimes slow a GKS application, compared to more device-specific software.

## APPLICATION OF THE GRAPHICAL KERNEL SYSTEM FOR SURFACE WATER PROGRAMS

For all five of the reasons listed in the previous section, GKS was selected to develop graphics output for surface-water programs. These programs include several rainfall-runoff and flow-routing models as well as programs for flow-duration analysis, frequency analysis and generalized leasts squares. These programs are used throughout the Geological Survey on Prime's, UNIX-based workstations, and personal computers. Thus, portability is of primary interest and can provide major savings in the resources required to develop and maintain the necessary graphics software.

As stated earlier, one of the major disadvantages of GKS is the additional programming effort required. Thus, one goal for the surface-water graphics software was the development of a set of graphics subroutines that would reduce the level of programming for subsequent applications. The use of a common set of subroutines also reduces the effort required to maintain the software. A brief description of the subroutines and graphics products is contained in the following sections. These subroutines are a first effort at establishing a standard set of graphics subroutines. When sufficient experience has been gained, the subroutines will be refined, redesigned, or discarded. The goal is to provide several layers of graphics subroutines for use in surface-water programs. At the highest layer would be several subroutines with a minimum number of arguments. At the lowest layer would be the GKS subroutines. Layers in between would include Fortran subroutines that perform part of the graphic display such as a log axis, time axis, or explanation text. At the top levels, many items are defaulted. At the lower layers, the programmer has to set more items or request them from the user. When a change is made to another graphics standard, in theory, only the lower layer routines need to be changed.

## Graphical Kernel System Subroutines

Table D1 lists the GKS subroutines that are used. A basic implementation of GKS is used with no segmentation or bundling. Only one workstation is open at a time.

## Graphics Subroutines Usage

Application programmers can use the graphics subroutines without learning the details of GKS. Programming with these subroutines is similar to using DISSPLA or PLOT10. All the subroutines are in a library that may be obtained from the U.S. Geological Survey, Office of Surface Water, 415 National Center, Reston, Virginia 22092.

The subroutines were designed around two common blocks, one for the data to be plotted and one for all the text and specifications for the plot (table D2). When all the appropriate variables in the common blocks have been set, a routine is called to make the plot.

The graphics subroutines can be used with one of three approaches:

(1) linking selected subroutines in the users' program,

(2) filling the common block using a subset of the subroutines and linking the subroutine library, or

(3) partially filling the common block using selected subroutines, then letting the user interactively set or change variables with an additional set of subroutines.

The third approach uses the most subroutines from the library. All the subroutines are defined in a system documentation file also available from the Office of Surface Water. That document is computer generated to help guarantee accuracy and more easily generate current documentation. For each subroutine, the document includes name, type, and purpose. Each argument is defined, the type is specified, and each is classified as input, modify, or output. Common blocks and variables that are used are also listed.

The variables used in the common blocks are listed and defined in table D2. Note, the maximum number of values to be plotted is 6,000 and the maximum number of variables is 18. Each of those can be changed in a PARAMETER statement in the include file for common blocks.

Approach 1 -- A few of the subroutines do not use the common block, they scale the axes, physically size the axes and draw the axes. A list of these subroutines is found in table D3. Arguments for these routines are described in the system documentation. As many as 80 characters can be used in the axes titles.

Approach 2 -- If the subroutine library is used, the common blocks cannot be included in the users' software. For this case, the set of utilities listed in table D4 are used to pass the data to the common blocks. Then utilities GPSTUP and PLTONE are called to make the plot.

Table D1.--Graphical Kernel System subroutines used in utilities

| FORTRAN 77 Name | GKS Function |
|---|---|
| GOPKS | OPEN GKS |
| GCLKS | CLOSE GKS |
| GOPWK | OPEN WORKSTATION |
| GCLWK | CLOSE WORKSTATION |
| GACWK | ACTIVATE WORKSTATION |
| GDAWK | DEACTIVATE WORKSTATION |
| GSWN | SET WINDOW |
| GSVP | SET VIEWPORT |
| GSWKWN | SET WORKSTATION WINDOW |
| GSWKVP | SET WORKSTATION VIEWPORT |
| GTX | TEXT |
| GFA | FILL AREA |
| GPL | POLYLINE |
| GPM | POLYMARKER |
| GSELNT | SELECT NORMALIZATION TRANSFORMATION |
| GQOPS | INQUIRE OPERATING STATE VALUE |
| GQDSP | INQUIRE DISPLAY SPACE SIZE |
| GQCF | INQUIRE COLOR FACILITIES |
| GQTXFP | INQUIRE TEXT FONT AND PRECISION |
| GQCHXP | INQUIRE CHARACTER EXPANSION FACTOR |
| GQCHSP | INQUIRE CHARACTER SPACING |
| GQCHH | INQUIRE CHARACTER HEIGHT |
| GQCHB | INQUIRE CHARACTER BASE VECTOR |
| GQTXX | INQUIRE TEXT EXTENT |
| GSASF | SET ASPECT SOURCE FLAGS |
| GSTXFP | SET TEXT FONT AND PRECISION |
| GSCHXP | SET CHARACTER EXPANSION FACTOR |
| GSCHSP | SET CHARACTER SPACING |
| GSTXAL | SET TEXT ALIGNMENT |
| GSCHH | SET CHARACTER HEIGHT |
| GSCHU | SET CHARACTER UP VECTOR |
| GSTXCI | SET TEXT COLOR INDEX |
| GSTXP | SET TEXT PATH |
| GSFACI | SET FILL AREA COLOR INDEX |
| GSFAIS | SET FILL AREA INTERIOR STYLE |
| GSFASI | SET FILL AREA STYLE INDEX |
| GSLN | SET LINETYPE |
| GSPLCI | SET POLYLINE COLOR INDEX |
| GSMKSC | SET MARKER SIZE SCALE FACTOR |
| GSPMCI | SET POLYMARKER COLOR INDEX |
| GSMK | SET MARKER TYPE |

Table D2.--Variables in common blocks

COMMON block CPLOTB.INC

| | |
|---|---|
| YX(n) | -values of data to be plotted (1 < n<6000) |
| BUFPOS(j,k) | -j =1,6 k=1,18<br>-for time-series start (j=1)/end(j=2) positions in YX array for each time-series or variable plotted (positions 3,4 not needed).<br>for x-y plots positions (j=1) and (j=2) for Y axis, positions (j=3,j=4) for X-axis<br>for x-y plots positions (j=5,j=6) for size of symbol if CTYPE = 7 |

COMMON block CPLOT.INC

| | |
|---|---|
| DEVCOD | -device code, system dependent number |
| DEVTYP | -output device category<br>1-screen,2-printer(impact and laser),3-plotter<br>4-GKS meta file, 5-DISSPLA meta file |
| FE | -Fortran unit number for GKS error file |
| NCRV | -number of curves |
| NVAR | -number of variables |
| PLMX(1) | -maximum value for Y-axis. |
| PLMN(1) | -minimum value for Y-axis. |
| PLMN(2) | -maximum value for Y-axis on right side |
| PLMN(2) | -minimum value for Y-axis on right side |
| PLMX(3) | -maximum value for auxilary axis. |
| PLMN(3) | -minimum value for auxilary axis. |
| PLMN(4) | -maximum value for X-axis |
| PLMN(4) | -minimum value for X-axis |
| YMIN(k) | -k=1,18 minimum value for each variable |
| YMAX(k) | -k=1,18 maximum value for each variable |
| TICS(k) | - number of tics on axes (default=10 except auxilary axis=2)<br>k=1 for Y-axis on left<br>k=2 for Y-axis on right<br>k=3 for auxilary axis<br>k=4 for X-axis |
| XTYPE | -type of X-axis<br>0-time<br>1-arithmetic<br>2-logarithmetic<br>3-probability percent (normal distribution) 99-1<br>4-recurrence interval (normal distribution) 1-100<br>5-probability fraction (normal distribution) .99-.01<br>6-probability percent (normal distribution) 1-99<br>7-recurrence interval (normal distribution) 100-1<br>8-probability fraction (normal distribution) 0.01-.99 |
| YTYPE(i) | -type of Y axis (i=1 for left axis) (i=2 for right axis)<br>0-none (applies only to right axis, left axis must be non-zero) |

```
                1-arithmetic
                2-logarithmetic
WHICH(k)        -which axis for each variable (k=1,NVAR)
                1-left        y-axis
                2-right      y-axis
                3-auxilary
                4-x-axis
CTYPE(k)        -type of curve (k=1,NCRV)
                1-uniform time step with lines or symbols
                  (main plot)
                2-uniform time-step with bars (main plot)
                3-uniform time-step with lines or symbols
                   (auxilary plot on top)
                4-uniform time-step with bars (auxilary plot on top)
                5-non-uniform (date-tagged) time-series
                6-x-y plot
                7-X-Y plot with symbol sized on a third variable
DTYPE           -data type for time-series
                  0-mean or sum over time-step
                  1-instantaneous or point data
TITL(24)        -title for the plot.
YLABL(80)       -label for the y-axis.
YXLABL(80)      -label for other axis (XTYPE=2 or YTYPE(2) =0)
YALABL(80)      -label for auxilary plot on top
LBC(j,k)        -j=1,18 k=1,NCRV
                -label for the k-th curve (20 characters)
LBV(j,k)        -j=1, 18, k=1,NVAR
                -label for the k-th variable (20 characters)
TSTEP(k)        -time step for each curve in TUNITS (k), k=1,NCRV
TUNITS(k)       -time units for each curve (2-min,4-day,5-mo,6-yr) k=1,NCRV
SDATIM(6)       -starting year,month,day,hour,minute, second of plot.
EDATIM(6)       -ending year,month,day, hour,minute, second of plot.
SYMBL(k)        -code for symbol type (k=1,NCRV)
                SYMBOL                      GKS CODE
                ------------                --------------
                NONE                        0
                .                           1
                +                           2
                *                           3
                O                           4
                X                           5

LNTYP(k)        -code for type of line (k=1,NCRV)
                LINE                        GKS CODE
                -------                     --------------
                NONE                        0
                SOLID                       1
                DASH                        2
                DOT                         3
                DOT-DASH                    4
```

PATTRN(k) -Code for shading (k=1,NCRV)

| PATTERN | GKS CODE |
|---|---|
| NONE | 1 |
| SOLID | 2 |
| HORIZ | 3 |
| VERT | 4 |
| DIAGONAL | 5 |

COLOR(k) -code for color (k=1,NCRV)

| COLOR | GKS CODE |
|---|---|
| background | 0 |
| B/W | 1 |
| RED | 2 |
| GREEN | 3 |
| BLUE | 4 |
| CYAN | 5 |
| MAGENTA | 6 |
| YELLOW | 7 |

BCOLOR -background color code number (0=white, 1=black)

YLEN -length of y-axis in world coordinates (WC) or both main and auxilary axis plus small space between them.

XLEN -length of x-axis (WC)

ALEN -auxilary plot axis length (WC)

YPAGE -vertical page size (WC).

XPAGE -horizontal page size (WC).

XPHYS -physical origin (WC) in horizontal

YPHYS -physical origin (WC) in vertical

SIZEL -height of lettering (WC)

LOCLGD(j) -action for legend (j=1,2)
-2.0=no legend
-1.0=legend in upper left corner (default location)
x,y=legend at fraction x and y of XLEN and YLEN from origin. (values between 0.0 and 1.0)

TRANSF(j,k) -transformation type for each variable (k=1,NVAR)
(j=1 for left main y-axis)
(j=2 for right y-axis or x-axis)
0=none
1=arithmetic (no transformation)
2=logarithmic
3=Normal distribution

BVALFG(4) -bad value flag for bottom,top,left,right
1=clip,plot at point going off scale
2=ignore, leave blank
3=plot arrow pointing off scale, don't connect lines
4=ignore,connect good points

BLNKIT(4) -min-max on y-axis and min-max on x-axis for box for no plotting (fractions from 0.0 to 1.0)

CTXT(i) -text to be placed on the plot (max 120 characters)

CPR -characters per line

```
NCHR        -number of characters to use (up to 120)
FYT         -fraction (0.0-1.0) of YLEN for upper left corner of text
FXT         -fraction (0.0-1.0) of XLEN for upper left corner of text
```

Table D3.--Subroutines not requiring common blocks

| | |
|---|---|
| AXAXIS - | draws arithmetic X-axis |
| AYAXIS - | draws arithmetic Y-axis |
| TMAXIS - | draws time axis based on start/end dates |
| PBAXIS - | draws probability X-axis in fraction, percent or recurrence interval |
| LXAXIS - | draws logarithmic X-axis |
| LYAXIS - | draws logarithmic Y-axis |
| SCALIT - | scales axes |
| SIZAXE - | sets origin, axes lengths and letter size based on size of plotting surface |

Table D4.--Subroutines to fill common block

| | |
|---|---|
| GPINIT - | initializes both common blocks |
| GPNCRV - | sets number of variables and curves |
| GPDATR - | adds data for a variable |
| GPTIME - | sets starting and ending time and time step and time units for each variable |
| GPLABL - | sets types of axes and titles |
| GPCURV - | sets symbol, color, line type, pattern and label for each curve |
| GPLEDG - | sets upper left corner for legend |
| GPTEXT - | sets supplementary text information to be printed on the plot |
| GPDEVC - | sets code for device type |
| GPSIZE - | set plot space, location of origin, axes lengths, and letter size |
| GPBLNK - | sets window for no graphics |
| GPVAR - | sets minimum and maximum for each variable and which axes for each variable |
| GPSCLE - | sets axes scales, number of tics, and action to be taken when plotting off scale |
| GPGNRL - | sets color of background |
| | GPERFL - sets Fortran unit number for GKS error file |

Approach 3 -- At this level, the applications programmer can use additional utilities in the library. This will let the user interactively modify variables pre-selected by the programmer such as symbol, line type, color, axes titles, or axes scales. Approach 3 is similar to Approach 2 except a main program shell needs to be modified and used to initialize and set up the system. At this level, a user configuration file TERM.DAT, can be used to set device-dependent attributes such as font, text precision, symbols, line types, fill patterns, and colors. After some of the variables in the common blocks have been set, the programmer calls the subroutine PROPLT which has arguments to indicate which of the following groups of variables (1) can not be changed, (2) can be changed and (3) need to be changed:

- data
- device
- axes
- curve specifications
- titles
- extras
- scales
- sizes.

At this point, control is turned over to the user to plot, modify, and re-plot until the user is finished and control is passed back to the program that called PROPLT.

## Use of the Program ANNIE for Graphics

Each of the types of plots can be created using the interactive program ANNIE (Lumb, and others, 1990). ANNIE is a computer program for interactive hydrologic analysis and data management. Data can be entered from one of three types of files, Watershed Data Management (WDM) file, PLTGEN file and flat file. Data also can be entered from the terminal. The WDM file is a binary, direct access file designed for storage and retrieval of data for surface-water application programs. The PLTGEN file is a sequential, formatted file for time-series data. The flat file or terminal input lets the user specify the starting position and length of each variable on the records in the file. Alternately, the file or terminal input can contain all the values for the first variable in free field format followed by all the values for the next variable in free field format, and so forth. For free field format, the values need to be separated by blanks or commas. The values can be on one or more records with a maximum record length of 132 bytes. Each variable in the free field format must start a new record. All of the other specifications for the plot are provided by way of menus.

Examples of ANNIE graphics are shown in figures D9 and D10. The first is a probability plot that illustrates the use of additional text. The second is a time-series plot of streamflow, evaporation, and precipitation for the 1957 water year. Precipitation is plotted on an auxiliary plot that is an option with time-series plots. The axis on the auxiliary plot is limited to arithmetic. The curve for time-series data was plotted as a step function for mean values, and was plotted as a line connecting points for instantaneous values.

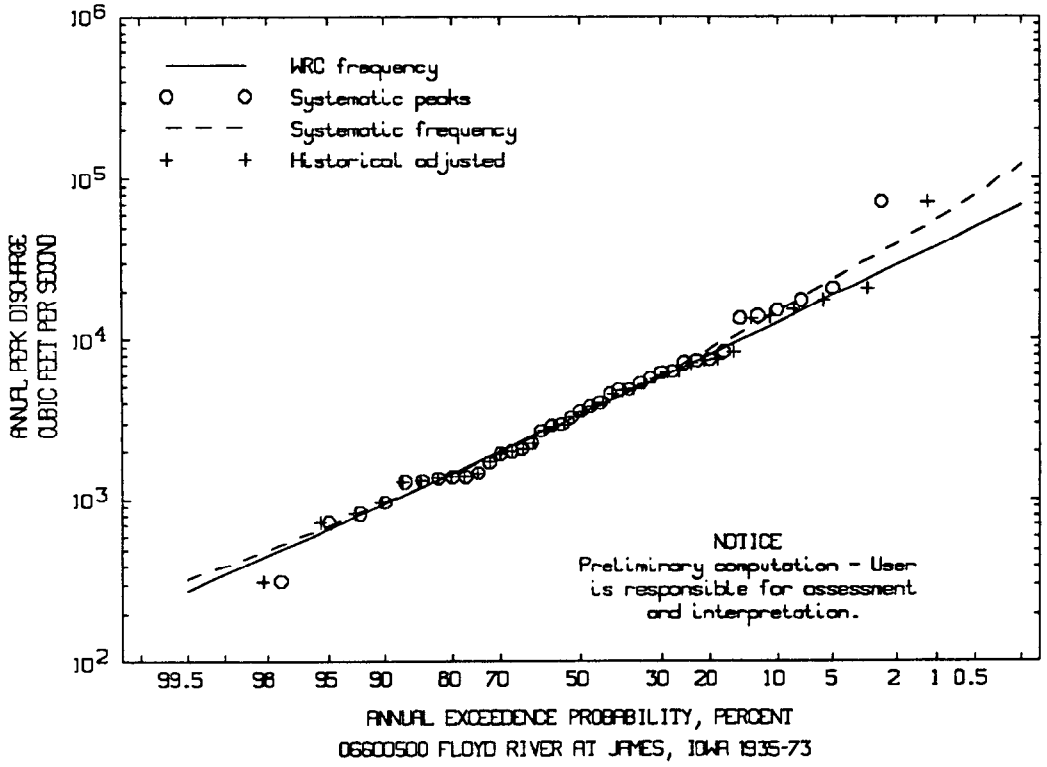


Figure D9.--Example of probability plot.

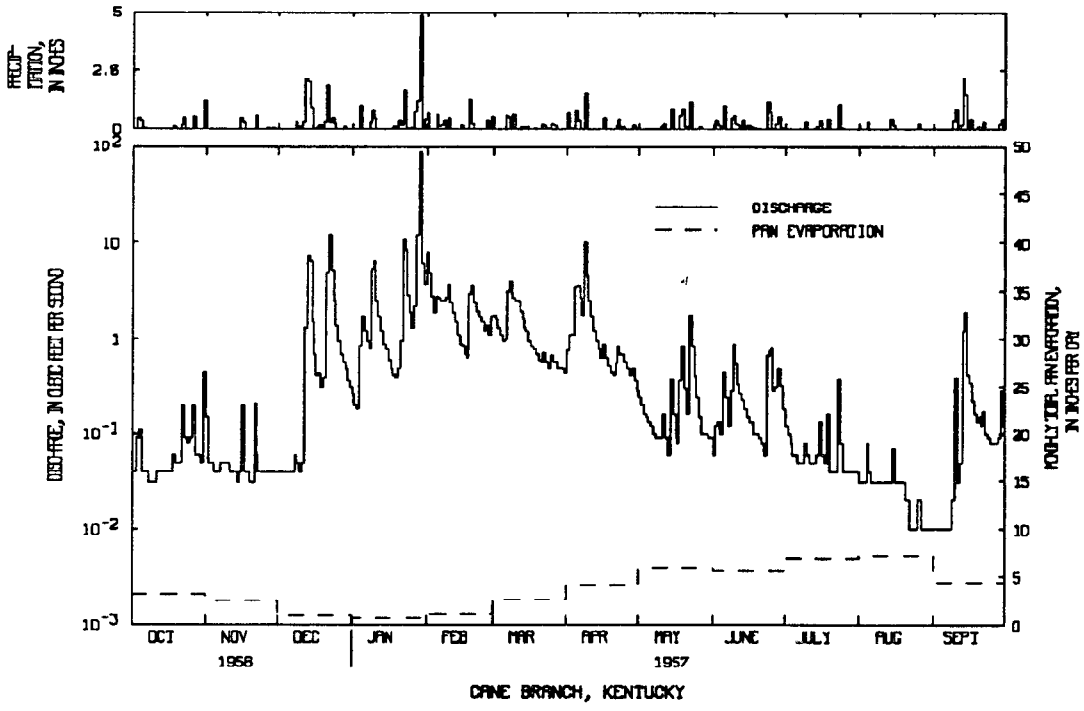


Figure D10.--Example of time-series plot.

## EXPERIENCE WITH THE GRAPHICAL KERNEL SYSTEM

GKS was not easy to learn. Although training courses are available from vendors, none were used. Two textbooks, one by Hopgood, 1983, and the other by Enderle, 1987, were quite helpful.

The Fortran binding for GKS is at a basic level. Fortunately, the previous graphics subroutine library for ANNIE was implemented at a basic level in an attempt to meet publication standards. Thus, the translation was fairly straightforward for drawing lines, axes, and writing text. Most of the difficulty was experienced in learning how to use the transformation routines from the world coordinates, to normalized device coordinates, to device coordinates. Locating text on the plot was initially confusing. Additional text problems centered around implementation of GKS.

Currently, the biggest disadvantage of GKS on the Prime is the inadequate implementation that is used. Other vendors are being evaluated as a potential source. Implementation of GKS on the personal computer is quite good and one vendor has drivers for over 120 devices. The Geological Survey has purchased the development library and a run-time distribution license for personal computers.

## SUMMARY AND CONCLUSIONS

During the 1970's, a growing demand for computer graphics programming standards emerged. Numerous standards organizations around the world began developing ideas for graphics standards. During the mid 1980's, the Graphical Kernel System was adopted as an international, American National, and Federal Government standard for computer-graphics programming.

The Graphical Kernel System is a hardware-independent, language-independent device-independent application interface for two-dimensional output and interactive input, providing maximum portability of applications. The Graphical Kernel System addresses the main functional areas of computer graphics programming. The main components of the Graphical Kernel System are output primitives, coordinate systems, segmentation, interactive input, logical workstation, and metafile generation.

The Graphical Kernel System is defined in a language-independent manner that specifies abstract routines. These abstract routines have been translated into standard language specific routines known as language bindings. The Graphical Kernel System has been implemented by numerous hardware and software vendors on most platforms, from microcomputers to supercomputers. The implementations most useful to the U.S. Geological Survey are those on Prime and UNIX-based workstations.

The Graphical Kernel System has been used to implement graphics software for surface-water applications. The disadvantages are the inadequate implementations by vendors and the limited number of device drivers. These disadvantages will be reduced as vendors continue to improve their implementations. The Graphical Kernel System was difficult to learn, but

other developers in the Geological Survey can use the utilities instead of the Graphical Kernel System directly.

The disadvantages are outweighted by the major advantage of portability. With the expense of developing and maintaining software, a common software package for Prime's, UNIX-based workstations, and personal computers provides a major savings in resources.

## REFERENCES CITED

American National Standard for Information Systems, 1985, ANSI X3.124 Computer Graphics-Graphical Kernel System (GKS) Functional Description, 268 p.

Enderle, G., Kansy, K., and Pfaff, G., 1987, Computer Graphics Programming. GKS - The Graphics Standard: New York, New York, Springer-Verlag, 632 p.

Federal Information Processing Standard, 1986, FIPS PUB 120 Graphical Kernel System (GKS), 3 p.

Hopgood, F. R. A., Duce, D. A., Gallop, J. R., and Sutcliffe, D. C., 1983, Introduction to the Graphical Kernel System: Orlando, Florida, Academic Press, Inc., 189 p.

Lumb, Alan M., Kittle, John L., Jr., and Flynn, Kathleen M., 1990, Users Manual for ANNIE, A Computer Program for Interactive Hydrologic Analysis and Data Management: WRI 89-4080, U.S. Geological Survey, 236 p.