

MEMORANDUM
RM-5531-ARPA
OCTOBER 1968

AD 677464

THE INTEGRATED GRAPHICS SYSTEM
FOR THE IBM 2250

Gary D. Brown and Charles H. Bush

DDC
RECEIVED
NOV 14 1968
RECEIVED
B

PREPARED FOR:
ADVANCED RESEARCH PROJECTS AGENCY

The **RAND** Corporation
SANTA MONICA • CALIFORNIA

Reproduced by the
CLEARINGHOUSE
for Federal Scientific & Technical
Information Springfield Va. 22151

MEMORANDUM
RM-5531-ARPA
OCTOBER 1968

**THE INTEGRATED GRAPHICS SYSTEM
FOR THE IBM 2250**

Gary D. Brown and Charles H. Bush

This research is supported by the Advanced Research Projects Agency under Contract No. DAHCl5 67 C 0111. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of ARPA.

DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

The **RAND** *Corporation*

1700 MAIN ST • SANTA MONICA • CALIFORNIA • 90406

This Rand Memorandum is presented as a competent treatment of the subject, worthy of publication. The Rand Corporation vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the authors.

Published by The RAND Corporation

PREFACE

The Integrated Graphics System (IGS) is a machine- and device-independent graphics system developed at The RAND Corporation. This Memorandum describes the IBM 2250 version of this system, which supports the RAND Tablet as an optional input device. IGS runs under Operating System 360 (OS/360) and permits the on-line user to interact with an IBM 360 computer through the 2250 graphic display console.

-v-

SUMMARY

This Memorandum describes the Integrated Graphics System (IGS) used at The RAND Corporation on an IBM 360 computer to support an IBM 2250 graphic display console and RAND Tablet. IGS runs under Operating System 360. Graphic applications may be programmed in a wide variety of languages, including FORTRAN, PL/I, SIMSCRIPT 1.5, and assembly language.

IGS is currently used with an IBM 2250. However, since IGS is independent of its specific hardware considerations, the system can easily be adapted to other graphic devices.

IGS is composed of subroutines that draw lines and circles, plot points, and display characters and numeric data. Subroutines are also provided to draw, label, and title graphs; manipulate displays; and request input from a graphic input device. IGS is specifically designed to be interactive, allowing the user to communicate directly with the computer and with his program in the computer through the IBM 2250 graphic display console and the RAND Tablet.

ACKNOWLEDGMENTS

The authors wish to thank D. T. Rumford of The RAND Corporation and M. J. Kaitz of Visual Computing Corporation for their help in designing the graphics system. We are also grateful to our RAND colleagues J. E. Rieber and R. A. Berman for programming assistance; G. F. Groner for writing the character-recognition routines; W. L. Sibley for his help in implementing these routines on the system; and Linda Colbert for her competent editorial assistance.

CONTENTS

PREFACE	iii
SUMMARY	v
ACKNOWLEDGMENTS	vii
FIGURES	xiii
Section	
I. INTRODUCTION	1
II. THE GRAPHICS SYSTEM HARDWARE	2
The 360 Model 40	2
The 2250 Display Unit	2
2250 Input Devices	4
The RAND Tablet	7
Hardcopy from the 2250	7
III. DESCRIPTION OF THE GRAPHICS SYSTEM	8
Mode Sets	8
Subject and Object Space	9
Subroutine-Naming and -Coding Conventions	11
The Structure of the Graphics System	11
Housekeeping Subroutines	11
Graphic Output Subroutines	12
Graph Subroutines	12
Graphic Manipulation Subroutines	12
Graphic Input Subroutines	13
Program Control Subroutines	14
IV. DESCRIPTION OF THE GRAPHIC SUBROUTINES	16
Brief List of Graphic Subroutines	16
1) Housekeeping Subroutines	16
2) Graphic Output Subroutines	16
3) Graph Subroutines	17
4) Graphic Manipulation Subroutines	17
5) Graphic Input Subroutines	17
6) Program Control Subroutines	18
7) Miscellaneous Subroutines	18
Format of Graphic Subroutine Write-Ups	18
Purpose	19
Usage	19
Standard Calls	19
Special Calls	19
Modifications	19
Miscellaneous Information	19
Housekeeping Subroutines	20

MODESG:	Initializes the System	21
GETIDG:	Assigns an ID and Defines a Display	22
SUBJEG:	Sets Up Subject Space	23
OBJCTG:	Sets Up Object Space	24
SETSMG:	Sets Mode Values	26
GETSMG:	Retrieves Mode Values	27
RSETMG:	Resets all Default Values in Mode Set Array	28
EXITG:	Closes Down the Graphic Devices	29
Graphic Output Subroutines		30
CIRARG:	Draws a Circle or an Arc	31
LEGNDG:	Types Characters	33
LINESG:	Draws Joined Lines	36
MLTPLG:	Draws Multiple Line Segments ...	38
NUMBRG:	Displays Numeric Data	40
POINTG:	Plots Points	43
SEGMTG:	Draws Line Segments	45
TEXTG:	Types Characters	46
Graph Subroutines		47
GRIDG:	Draws a Grid	48
LABELG:	Labels a Grid	52
TITLEG:	Titles a Grid	58
SETUPG:	Computes Appropriate Arguments for the Grid Routines	62
GRAPHG:	Constructs a Complete Graph	65
Graphic Manipulation Subroutines		67
DISPLG:	Deletes or Turns Displays Off or On	68
GPCHRG:	Manipulates Characters in Displays	69
GPXYG:	Manipulates X,Y Coordinates within a Display	71
Graphic Input Subroutines		73
GSTATG:	Checks the Status of a Device ..	74
WAITSG:	Waits for Console Action	77
CHARG:	Inputs Characters with the RAND Tablet	80
DATAG:	Inputs Data	82
RECOG:	Recognizes and Displays Geometric Figures	85
Program Control Subroutines		87
AREASG:	Sets Up Rectangular Control Areas on the 2250 Scope	88
BUTTNG:	Sets Up Control Buttons on the 2250	90
PUSHG:	Determines Whether a Control Area is Pushed	92
Miscellaneous Special-Function Subroutines		93
ALARMG:	Sounds the 2250 Audio Alarm	94
CHANG:	Identifies the IDs that have had Character Changes	95
CONVTG:	Converts Characters to Numeric .	96

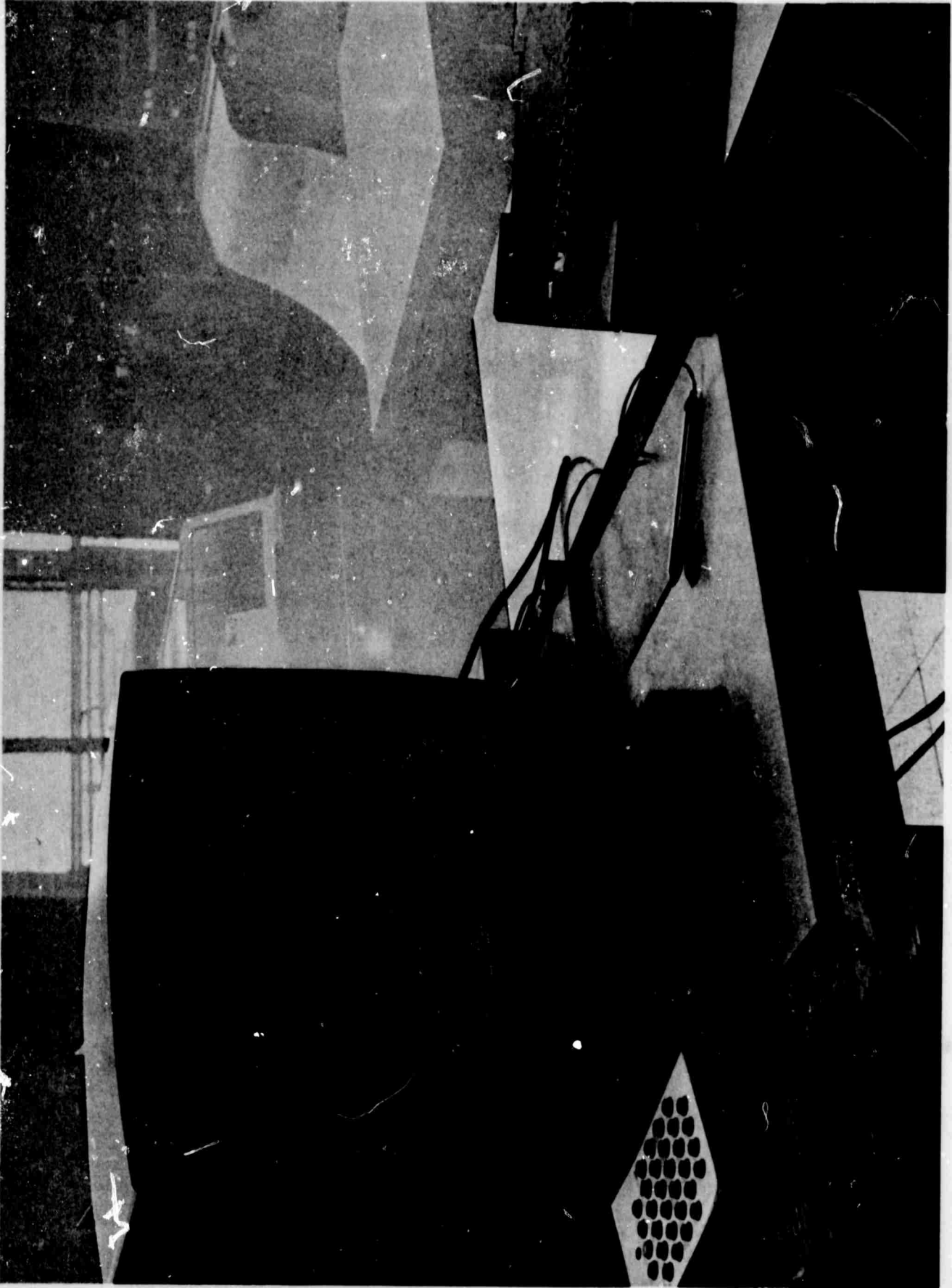
CURSRG:	Inserts or Removes Cursor	97
DEBUG:	Debugging Aid	99
FMTSG:	Converts Numeric Data to Characters	100
GTIMEG:	Returns the Time of Day	101
SETFKG:	Sets the Function Keys	102
SNAPG:	2250 Hardcopy	104
WTIMEG:	Pauses for a Specified Length of Time	105

Appendix

A.	SAMPLE PROGRAMS	107
B.	IGS ERROR CODES	117
C.	SUMMARY OF 2250 GRAPHIC FEATURES	118
D.	CORE STORAGE ALLOCATION	119
E.	SUBROUTINE CROSS-REFERENCE LIST	121
F.	JOB CONTROL LANGUAGE	123
G.	PL/I COMPATIBILITY	125
H.	IGS MODE SET ARRAY	127
I.	HARDCOPY REQUESTS	131
J.	SYSTEM SUBROUTINES	133
	BUFFZZ: Changes the Internal Buffer Size	134
	CBUTZZ: Creates, Then Files, a Control Entry	135
	CHARZZ: Recognizes Handwritten Characters	136
	CTABZZ: Turns the RAND Tablet On and Off	139
	C225ZZ: Turns the IBM 2250 On and Off ...	140
	ERRZZ: General Error Subroutine	141
	FBUTZZ: Detects a Button Touch	142
	GETCZZ and	
	PUTCZZ: Pack or Unpack Bytes	143
	GNXYZZ: Computes the Next X,Y Location of a Button	144
	IFMZZ: Converts an Integer Number into a Character String	145
	METAZZ: Writes Graphic Commands	147
	PACKZZ: Packs Bytes into the Output Buffer	149
	PZZ: Prints Out the Mode Set Array ...	151
	REPKZZ: Repacks the 2250 Buffer	152
	SCALZZ: Converts User Coordinates to Rasters	153
	TABLZZ: Resident Data Section	155
	TRAPZZ: Processes 2250 Input Traps	156
	UNSCZZ: Converts Rasters to User Coordinates	157
	REFERENCES	159
	ALPHABETIC LIST OF SUBROUTINES	161

FIGURES

The IBM 2250 Display Station with RAND Tablet	xiv
1. The Character Envelope	5
2. Subject and Object Space	10
3. Graphic Output Subroutines	30
4. Graph Subroutines	47
5. GRIDG Subroutine	49
6. LABELG Subroutine	53
7. TITLEG Subroutine	59
8. Graphic Manipulation Subroutines	67
9. Program Control Subroutines	87
10. Output from Sample Program 1	108
11. Output from Sample Program 3	115



The IBM 2250 Display Station with RAND Tablet

I. INTRODUCTION

This Memorandum describes the Integrated Graphics System (IGS) currently used at The RAND Corporation. The system now supports a single Model 1 IBM 2250 display unit running on a 360 Model 40, but could easily be expanded to handle multiple 2250s. The RAND Tablet [1] is an optional input device. The system uses basic programming support for the 2250 in the OS/360 operating system.

IGS consists of subroutines called by the user to perform such graphic functions as creating, deleting, moving, or replacing displays or parts of displays, and requesting input from a graphic input device. The user may code in a variety of languages--FORTRAN, PL/I, assembly language, SIMSCRIPT 1.5, or any other language with standard OS/360 linkages. IGS relieves the user from the intricate, specific hardware considerations of the 2250, allowing him to concentrate on his displays.

IGS is specifically designed to be interactive, allowing the user to communicate directly with the computer and with his program in the computer through the IBM 2250 graphic display console and the RAND Tablet.

II. THE GRAPHICS SYSTEM HARDWARE

THE 360 MODEL 40

IGS currently runs on a 256,000-byte 360 Model 40. In 360 terminology, memory size is given in terms of bytes and not words, with 8 bits per byte, and 4 bytes per word. The machine now operates with version 14 of OS/360, which occupies 20,512 bytes or 5128 words of core.

Along with the basic 360/40, the system includes one card reader and punch, one printer, two 2260 display units, one Model 1 2250, four 2311 disk drives, one 1052 console typewriter, one RAND Tablet, one 2701 data channel for the RAND Tablet, and limited access to one 7-track and one 9-track tape normally part of the 360/30.

IGS will run on any IBM 360 computer large enough to support OS/360. The 2701 data channel is necessary only if the RAND Tablet is included as an input device. IGS will operate without the RAND Tablet.

THE 2250 DISPLAY UNIT

The 2250 display unit consists of a cathode ray tube (CRT) on which images are displayed under programmed control, and input devices (a light pen, an alphameric keyboard, and a programmed-function keyboard) for entering data from the 2250 console into the computer. The 2250 generates displays from its own external buffer requiring no computer time or core storage. Displays are created when the graphics system moves certain bit configurations representing graphic commands into the external buffer.

The 2250 buffer is analogous to a small computer, with the graphic commands in the buffer representing the computer instructions. The 2250 graphic commands direct a small beam of light across the screen, drawing the picture. The beam may be either on (unblanked) or off (blanked) while it

is being moved. However, images appear only if the beam is moved in the unblanked mode. Because the images fade rapidly, a steady and stationary display requires constant image regeneration. This is accomplished by repeating the execution of the orders and data in the 2250 buffer thirty to forty times per second. The actual regeneration rate is a function of the amount of data displayed. Excessive data will cause the screen to flicker because regeneration is not fast enough to preserve the image.

The character generator, a special hardware feature in the 2250, directs the beam to draw characters. Decoding these instructions in the buffer makes the 2250 considerably more complex (and more expensive) than a TV set. The 2250 buffer operates independently of the main 360 computer. The 360 computer can only start or stop the buffer and read or write into it.

The user need not concern himself with the 2250 buffer, since IGS generates all the 2250 graphic commands needed and reads or writes the buffer as needed.

The usable portion of the CRT screen (12 × 12 in.) is defined by a matrix (1024,1024) of addressable point positions. However, the points are actually addressed from 0 to 4095, with every fourth point being addressable. The distance between any two addressable points (in either the x or y direction) on the display area of the CRT is defined as a raster unit.

The origin (coordinate 0,0) begins at the lower left corner of the screen, then extends horizontally to the right in the x-direction and vertically in the y-direction, so that the coordinates (4095,4095) at the top right corner represent the maximum boundary of the screen.

Alphameric characters may be created by using the character generator or by drawing the desired characters with a series of lines. The character generator produces characters of two sizes (normal and one-and-one-half times

normal) and of one orientation (vertical). The center of a character is its x,y location. Figure 1 illustrates the details of character dimensions. Appendix C (p. 118) describes the available characters and their sizes.

The 2250 scope produces three basic types of displays: vectors, points, and characters. Vectors may be drawn between any two addressable points on the 2250 screen. Points and characters may appear at any addressable point on the screen.

The 2250 displays two types of characters--protected and unprotected. The user may replace unprotected characters by typing over them with the alphameric keyboard or writing over them with the RAND Tablet pen; protected characters are inalterable. Two corresponding types of vectors or points have been defined for IGS: protected vectors and points, which cannot be moved on the scope; unprotected vectors and points, which can be moved around by special subroutines.

The 2250 buffer size is 8192 bytes. An x,y coordinate occupies four bytes, a character one byte, and a delimiter indicating the type of output fills two bytes. Reference 2 gives the complete details of the 2250 buffer commands. The 2250 also has an audio alarm, an unpleasant whistle.

2250 INPUT DEVICES

The 2250 responds to three types of graphic input devices--a light pen, an alphameric keyboard, and a programmed-function keyboard.

The light pen is a pointing device. It detects the light from the 2250 beam when the beam enters the pen's field of vision. The user, by pointing with the light pen at a particular display and then pressing a foot pedal, interrupts the computer, which reads in the x,y coordinates of the display as well as their location in the buffer.

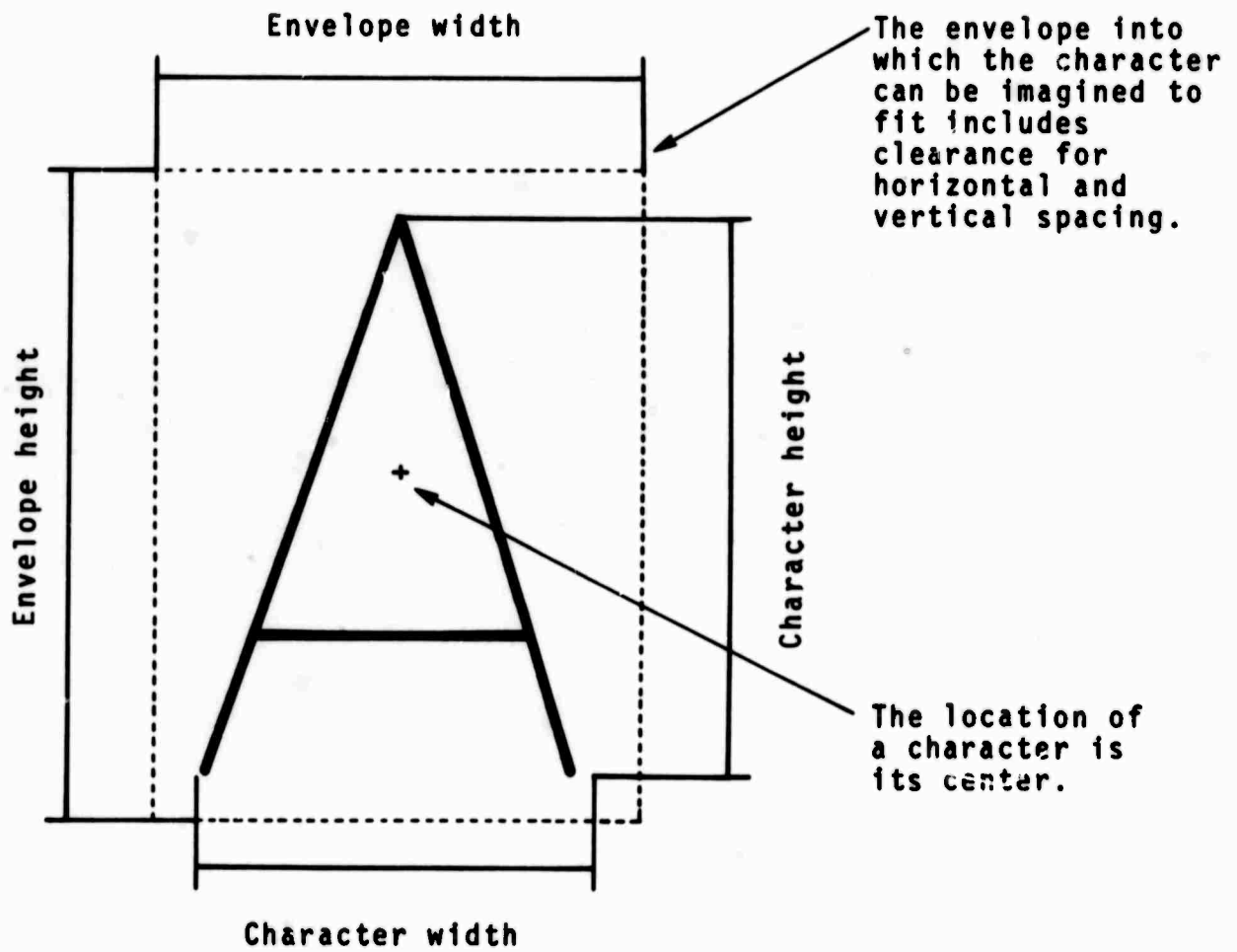


Fig. 1--The Character Envelope

The resolution of the light pen is poor because of a parralax effect caused by the thickness of the glass in the 2250 screen, and because the user's hand tends to obscure the image at which the user is pointing. The light pen can detect only displayed images; blank areas on the screen can not be detected.

The alphameric keyboard, a typewriter-like device, enters information directly into the 2250 buffer. It can type over only unprotected characters after a cursor has been positioned under a character. The cursor, which specifies the location of the next typed character, appears on the scope as a dash under that character. It can be inserted only by software, but once it appears on the screen, the user may position it by spacing forward or backward, or jumping to another line. The cursor jumps from line to line in the order in which the lines were originally displayed. This may be unrelated to their order on the screen.

New characters can be typed in only by overstriking previously displayed characters (which may be blanks). New characters replace the old characters directly in the 2250 buffer without computer assistance. Unfortunately, this means that the computer, and the user's program, has no easy way to determine what or where characters have been entered.

Striking the END or CANCEL key on the alphameric keyboard interrupts the computer. The user's program then determines which key was struck. The labels *END* or *CANCEL* on the keys have no inherent meaning; both keys interrupt the computer when struck.

The programmed-function keyboard is a separate box containing 32 keys, numbered 1 through 32. When a function key is struck, the computer is interrupted and notified of which key was struck. An overlay may be placed over the keys to indicate their function. The user may detect both which overlay is in place and which key was touched. Selected function keys may also be lit up. However, lighting up the keys neither locks nor unlocks them; it simply illuminates them.

THE RAND TABLET

The RAND Tablet may be used as a pointing device similar to the light pen, as a tracing device to input a series of points, as a character-input device similar to the alphameric keyboard, or as drawing device to input such geometric figures as rectangles and circles.

The RAND Tablet consists of a pen-like stylus and a special, flat, horizontal surface located in front of the vertical 2250 screen. The 10-in.-sq surface or Tablet is a photoetched grid of 1024 lines with a resolution of one hundredth of an inch in both the x and y directions. Pressing the pen on the Tablet closes a pressure-sensitive switch in its tip, thereby inputting its x,y coordinates. The user writes on the Tablet, but views his writing on the 2250 screen. A small point displayed on the screen shows the pen's position relative to the Tablet surface. The user quickly adjusts to this disjointed method of writing.

Special subroutines are provided to recognize characters handwritten on the Tablet [3,4]. When the user writes a character, "ink" in the form of short vectors follows the track of his pen. When he raises his pen, these routines attempt to recognize the handwritten character, then erase the "ink" and replace the handwritten character with the 2250-generated character.

HARDCOPY FROM THE 2250

Sizes 8 1/2 x 11 and 11 x 14 hardcopy for all 2250 IGS programs are obtained by reading the 2250 buffer into the 360 core, converting it into a form acceptable to the S-C 4060 [5], writing it onto disk and later onto tape, then processing the tape on the S-C 4060. The S-C 4060 is a high-quality graphic display device capable of producing film or hardcopy output. Appendix I (p. 131) describes the method for obtaining hardcopy. The S-C 4060 is necessary only if hardcopy is desired.

III. DESCRIPTION OF THE GRAPHICS SYSTEM

The graphics system consists of subroutines that draw lines, plot points, display characters, manipulate displays, and request input from a graphic device. These superficially simple functions are actually complex because of the many possible variations. For example, characters can be protected or unprotected, basic or large size. Mode sets allow the user to execute such simple actions as displaying characters without becoming confused by all the variations possible.

MODE SETS

Parameters are passed to IGS subroutines in two ways: as arguments in the subroutine calls and as values stored in the *mode set array*. The mode set array is defined as a 200-word, single-dimensioned, real array in the user's program and must appear as an argument in each call to an IGS subroutine. Default values are stored in this array when the system is initialized.

Those parameters that change with each call are included as arguments in the subroutine calls; all others are held in the mode set array. Limiting the number of arguments necessary in the subroutine calls facilitates their use.

For example, to display characters, the user need specify only the characters themselves, the x,y location of the first character, and the number of characters. However, the mode set array automatically provides many other default parameters--character size, line spacing, margins, line orientation, etc. The user alters such values by calling subroutine SETSMG to store a new value in the array. These alterations will remain in effect until the user changes them again.

SUBJECT AND OBJECT SPACE

IGS automatically scales from user coordinates to absolute rasters. (If the user wishes to address absolute rasters himself, he can order the system to do no conversion.) The user must specify two items of information before the system can set up scaling factors. The first item is the size of the user's *subject space*, i.e., the coordinate system that bounds his data. The system will assume it to be the same as the raster size of the display surface, 4096.0 × 4096.0, unless otherwise informed through a call to subroutine SUBJEG.

Subject space is useful in both plotting--where the data's physical units may not correspond to the raster address of the 2250--and textual displays. For example, by letting the x-axis range from 1 to 72 (the number of normalized characters per line) and the y-axis range from 1 to 52 (the number of lines per page), lines and character positions may be addressed directly, as on a typewriter.

The second item is the *object space*, i.e., the portion of physical display surface used. It is expressed in "normalized" units to differentiate it from raster units. A value of 1.0 is the smallest dimension of the display surface in either the x or y axis. For the 2250, this is 4096 raster units. The system will assume it to be the full display surface size, 1.0 × 1.0, unless otherwise informed through a call to subroutine OBJCTG. The ability to subdivide the display surface enables the user to put a margin around his picture, or to display multiple pictures simultaneously.

When an x or y coordinate is scaled into an area outside the physical screen, an error message is printed. The coordinate is set to zero, if it is below or to the left of the screen, and to 4095 if above or to the right.

Figure 2 illustrates the distinction between subject and object space.

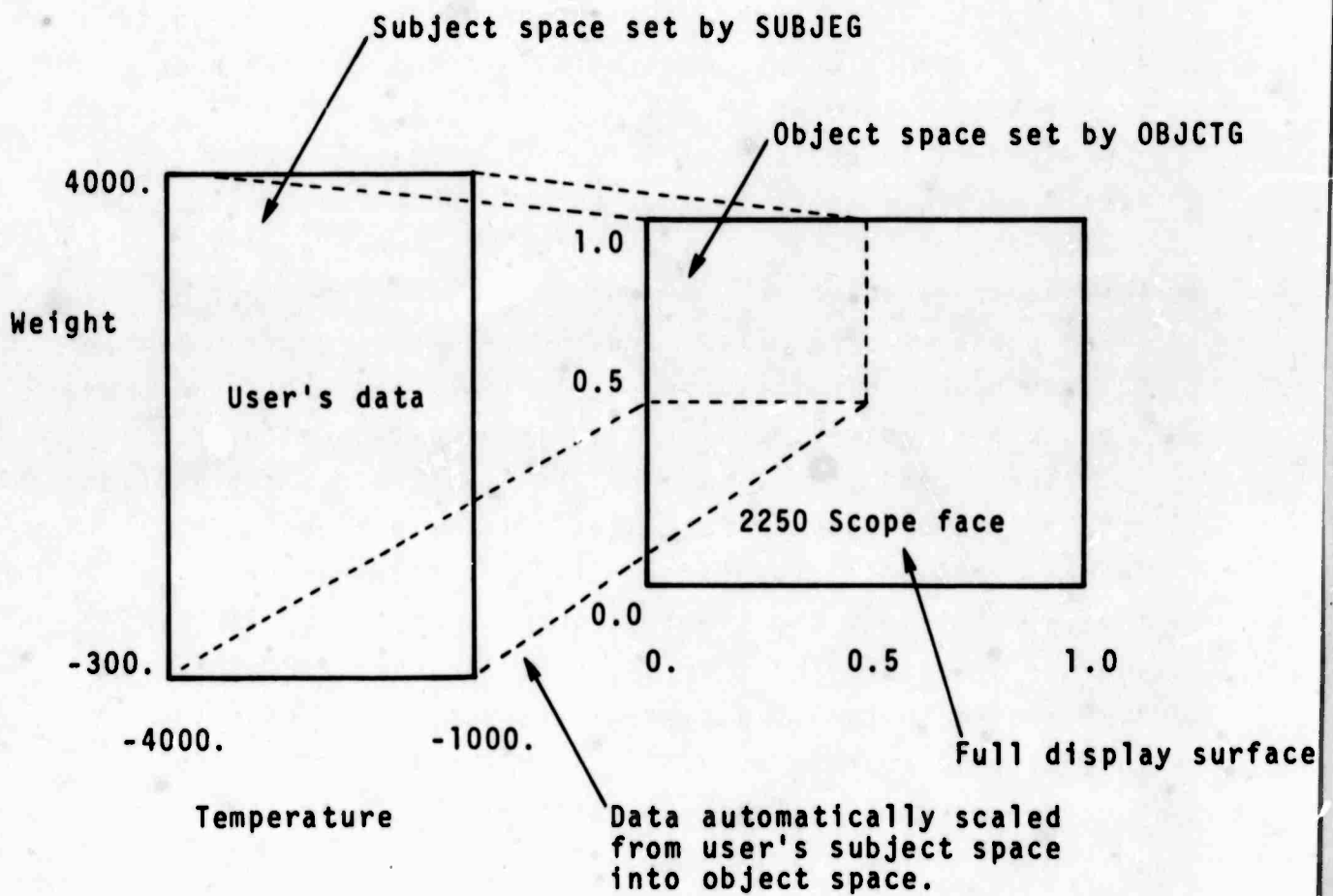


Fig. 2--Subject and Object Space

SUBROUTINE-NAMING AND -CODING CONVENTIONS

The following conventions have been adopted for the design and coding of IGS.

- 1) All general graphic subroutine names end with a G.
- 2) All system subroutine names not of general interest end with ZZ.
- 3) All function names end with a Z.
- 4) The first argument of each call to a graphic subroutine must be the mode set array defined in the user's program.
- 5) All variables in the mode set array are real.
- 6) Coordinate values in the calls to graphic subroutines are real.
- 7) Variable-length calls are not used unless one of the arguments contains the number of arguments in the call.
- 8) The subroutines are coded in Level-G FORTRAN where possible.
- 9) OS/360 is not modified.

THE STRUCTURE OF THE GRAPHICS SYSTEM

IGS is composed of several classes of subroutines: housekeeping, graphic output, graph, display manipulation, graphic input, program control, and miscellaneous.

Housekeeping Subroutines

The user starts the graphics system with a call to MODESG, and stops it with a call to EXITG. After starting the system, he may call SUBJEG to tell it what units his data will occupy; the system will then calculate the scaling factors needed. He may also call OBJCTG to specify the

portion of the 2250 screen he wants to use. This enables him to subdivide the screen so he has, in effect, more than one scope in front of him. Both calls are optional. SETSMG and GETSMG set and retrieve mode set values, respectively. RSETMG sets all modes to their default values.

The user must call GETIDG to ask the system for a number to identify displays before creating them. The system classifies all subsequent graphic output under this ID until another ID is requested. An ID usually identifies a complete display (containing characters, vectors, or points) but may also designate only a part of a display (e.g., a single character). The ID identifies the group of displays to be treated as an entity. Display organization is left up to the user.

Graphic Output Subroutines

Graphic output subroutines display lines (LINESG, SEGMTG); characters (LEGNDG, TEXTG); circles (CIRARG); points (POINTG); numbers (NUMBRG) in I, E, or F format; and multiple line segments (MLTPLG) for drawing grids or crosshatching.

Graph Subroutines

Subroutines draw (GRIDG), label (LABELG), and title (TITLEG) grids. GRAPHG plots an entire graph, complete with grid, labels, titles, and data. An additional subroutine (SETUPG) computes aesthetically pleasing grid and label parameters for GRIDG and LABELG.

Graphic Manipulation Subroutines

The user may call subroutines to manipulate his displays. DISPLG causes displays to disappear or reappear on the 2250 screen, or deletes them. Deleting a display not only causes it to disappear, but also releases all storage associated with that display.

The system allows the manipulation of the individual components of the displays, as well as the whole display. A user can retrieve or change any character within a display by calling GPCHRG, and retrieve, change, or move any x,y coordinate by calling GPXYG. Only unprotected x,y coordinates can be manipulated individually; protected x,y coordinates can not. This distinction saves core storage, because the unprotected x,y coordinates must be stored in main computer core to enable graphic manipulations fast enough to prevent flicker, while the unmanipulated protected coordinates need not be stored. A mode set call specifies protected or unprotected displays.

The user refers to characters and x,y coordinates within a display by a number corresponding to their order of creation. For example, he might ask for three characters from a display, starting at the seventh character, or he might order six x,y coordinates, starting with the second, to be moved a certain distance.

Graphic Input Subroutines

The graphic input subroutines allow the user to enter data and control the operation of his program from the 2250 console. Subroutine GSTATG checks the status of a specified graphic input device. Subroutine WAITSG suspends program operation until the user initiates some activity at the console. When the status of a 2250 input device is checked, one is essentially asking if that device has been used since the last time it was checked. Waiting for input from a graphic device suspends all activity in the user's program until some action is initiated from the console.

When return is made from GSTATG or WAITSG, the user is told which device was used, whether the end or cancel key was hit, which programmable-function key was hit, which overlay was in place, the x,y location of the pen and

which display, if any, it touched, or whether the RAND Tablet pen is up or down.

IGS does not permit the user's program to get control directly from an asynchronous interrupt caused by a graphic input device. Instead, the program may check the status of the graphic input devices at appropriate places, or specifically await some graphic input.

CHARG recognizes characters handwritten with the RAND Tablet and flags the ID of the appropriate display as having a character changed. CHARG permits only unprotected characters to be overwritten.

A higher-level graphic input subroutine (DATAG) allows the user to enter numeric data directly from the 2250 console with either the keyboard or the RAND Tablet. The user calls DATAG to pass over a list of variables to be changed. For each variable, he must give an alphameric name so it can be identified on the screen, the format of the variable (I, F, E, or A format), and the variable itself. DATAG then displays the current value of each listed variable on the 2250 screen beside the name, so the user may examine and alter the values. Striking the END key causes DATAG to erase the display and return.

RECOG recognizes geometric figures hand-drawn with the RAND Tablet. When the geometric figure is recognized, RECOG erases the hand-drawn figure and replaces it with an exact geometric figure.

Program Control Subroutines

Program control subroutines enable the user to control his program from the console. Subroutines BUTTNG and AREASG define sensitive character strings and areas on the 2250 screen. Subroutine PUSHG permits the user's program to detect when these character strings or areas are touched.

The sensitive character strings and areas are actually user-defined buttons. The user may build and label several buttons, then write his program to respond to a touch on a button.

BUTTNG displays any number of sensitized character strings at any place on the screen. AREASG defines any number of sensitized areas on the screen. An area may optionally have lines drawn around it to make it visible.

Subroutine PUSHG detects a button touch by either the light pen or RAND Tablet stylus. PUSHG displays a double asterisk beside a touched button to notify the user that it recognizes and has executed his request.

IV. DESCRIPTION OF THE GRAPHIC SUBROUTINES

BRIEF LIST OF GRAPHIC SUBROUTINES

The following subroutines constitute the graphics system. A complete description of each subroutine follows this outline.

1) Housekeeping Subroutines

- a) **MODESG** Initializes the graphics system. (This is the first graphic subroutine call.) (p. 21.)
- b) **GETIDG** Provides a number to identify a display (p. 22).
- c) **SUBJEG** Sets up the user's coordinate system (subject space) (p. 23).
- d) **OBJCTG** Sets up the portion of the 2250 screen to be used (object space) (pp. 24-25).
- e) **SETSMG** Stores a value in the mode set array (p. 26).
- f) **GETSMG** Retrieves a value from the mode set array (p. 27).
- g) **RSETMG** Resets all modes to their default values (p. 28).
- h) **EXITG** Terminates IGS. (This is the last call to the graphics system.) (p. 29.)

2) Graphic Output Subroutines

- a) **CIRARG** Draws circles or arcs (pp. 31-32).
- b) **LEGNDG** Displays characters or text at a given x,y location (pp. 33-35).
- c) **LINESG** Draws joined lines (pp. 36-37).
- d) **MLTPLG** Draws multiple line segments (pp. 38-39).

- e) NUMBRG Displays numeric data (pp. 40-42).
- f) POINTG Plots symbols or points (pp. 43-44).
- g) SEGMTG Draws noncontiguous line segments (p. 45).
- h) TEXTG Displays characters or text from current point (p. 46).

3) Graph Subroutines

- a) GRIDG Draws a grid (pp. 48-51).
- b) LABELG Labels a grid (pp. 52-57).
- c) TITLEG Titles a grid (pp. 58-61).
- d) SETUPG Computes aesthetically pleasing grid and label parameters (pp. 62-64).
- e) GRAPHG Draws, labels, and titles a complete graph (pp. 65-66).

4) Graphic Manipulation Subroutines

- a) DISPLG Deletes, turns on, or turns off displays (p. 68).
- b) GPCHRG Retrieves or replaces characters in a display (pp. 69-70).
- c) GPXYG Retrieves, replaces, or moves x,y coordinates within a display (pp. 71-72).

5) Graphic Input Subroutines

- a) GSTATG Checks the status of a graphic device (pp. 74-76).
- b) WAITSG Waits for graphic input (pp. 77-79).
- c) CHARG Inputs characters with the RAND Tablet (pp. 80-81).
- d) DATAG Inputs data from the 2250 console (pp. 82-84).

- e) RECOG Recognizes geometric figures hand-drawn with the RAND Tablet (pp. 85-86).

6) Program Control Subroutines

- a) AREASG Defines a sensitive area (pp. 88-89).
- b) BUTTNG Defines and displays buttons (pp. 90-91).
- c) PUSHG Determines if a sensitive area or button has been touched (p. 92).

7) Miscellaneous Subroutines

- a) ALARMG Sounds the 2250 audio alarm (p. 94).
- b) CHANG Determines which displays the RAND Tablet changed (p. 95).
- c) CONVTG Converts characters to numeric (p. 96).
- d) CURSRG Controls the 2250 cursor (pp. 97-98).
- e) DEBUG Displays selected variables on the 2250 screen (p. 99).
- f) FMTSG Converts numeric data to characters (p. 100).
- g) GTIMEG Determines the current time and day (p. 101).
- h) SETFKG Controls the 2250 function key lights (pp. 102-103).
- i) SNAPG Provides 2250 hardcopy (p. 104).
- j) WTIMEG Waits a specified length of time (p. 105).

FORMAT OF GRAPHIC SUBROUTINE WRITE-UPS

The graphic subroutine descriptions conform to the following format:

PURPOSE

Purpose of the subroutine.

USAGE

Standard Calls

Details of the subroutine calling sequence. FORTRAN notation indicates integer or real arguments. Arguments beginning with A through H or O through Z are real; those beginning with I through N are integer.

Underscoring an argument designates it as an input value to the called subroutine. The argument may be either a constant or a variable. Overscoring an argument designates it as an output variable returned from the called subroutine. The argument must be a variable. A line both above and below an argument indicates that the argument is both an input and an output. It must be a variable. For example,

```
CALL SUBR(A,B,C)
```

- 1) A will contain an input value for SUBR.
- 2) B will return with a value from SUBR.
- 3) C will contain an input value for SUBR, and will return with a value from SUBR.

Special Calls

Special calls to the subroutine.

Modifications

Modifications of the subroutine through mode sets.

MISCELLANEOUS INFORMATION

Language in which the subroutine is coded and any additional information not covered above.

HOUSEKEEPING SUBROUTINES

This section briefly describes the housekeeping subroutines. The arguments in the calling sequences are detailed in the individual subroutine descriptions.

```
C   THE USER FIRST DIMENSIONS THE MODE SET ARRAY AND
C   INITIALIZES THE SYSTEM BY CALLING MODESG BEFORE
C   REQUESTING GRAPHIC OUTPUT. MODESG SETS ALL DEFAULT
C   VALUES IN THE MODE SET ARRAY AND OPENS THE OUTPUT
C   FILES. THE MODE SET ARRAY MUST BE 200 WORDS LONG.
      DIMENSION Z (200)
      CALL MODESG (Z, 0)
C   HE MAY THEN OPTIONALLY CHANGE THE COORDINATE SYSTEM
C   BY CALLING SUBJEG TO CHANGE THE SUBJECT SPACE, OR
C   OBJCTG TO CHANGE THE OBJECT SPACE.
      CALL SUBJEG (Z, XMIN, YMIN, XMAX, YMAX)
      CALL OBJCTG (Z, XMIN, YMIN, XMAX, YMAX)
C   SETSMG MAY BE CALLED TO CHANGE THE DEFAULT MODE SET
C   VALUES.
      CALL SETSMG (Z, NO, VALUE)
C   GETIDG MUST BE CALLED BEFORE ANY DISPLAYS ARE CREATED.
C   GETIDG ASSIGNS A NUMBER TO IDENTIFY THE DISPLAYS
C   THAT FOLLOW.
      CALL GETIDG (Z, ID)
C   THE USER IS NOW READY TO CREATE DISPLAYS. TO RESET
C   THE MODE SET ARRAY TO ITS DEFAULT VALUES, HE MAY
C   MAKE EITHER INDIVIDUAL CALLS TO SETSMG OR A SINGLE
C   CALL TO RSETMG.
      CALL RSETMG (Z)
C   GETSMG RETRIEVES MODE SET VALUES.
      CALL GETSMG (Z, NO, VALUE)
C   EXITG TURNS OFF THE 2250. IT MUST BE CALLED BEFORE
C   THE JOB IS TERMINATED.
      CALL EXITG (Z)
C   THE USER TERMINATES HIS PROGRAM IN THE NORMAL MANNER.
      CALL EXIT
      END
```

MODESG: Initializes the System

PURPOSE

Initializes the graphics system, opens files, and sets all default values in the mode set array. It must be the first graphic subroutine called.

USAGE

Standard Calls

CALL MODESG(Z, IDEV)

- 1) Z must be a 200-word array defined in the user's program. All mode sets and system communication pointers are stored in this array.
- 2) IDEV specifies the devices to turn on.
If IDEV = 0, turn on the 2250 only.
= 1, turn on the 2250 and the Tablet.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

GETIDG: Assigns an ID and Defines a Display

PURPOSE

Defines the beginning of a new display, and returns a number uniquely identifying the display. All displays generated subsequent to this call, until terminated by another call to GETIDG, will be regarded as one logical display.

The first ID returned is a two. Each subsequent call returns an integer number one larger than the last. GETIDG must be called at least once before any displays are created.

USAGE

Standard Calls

CALL GETIDG(Z,ID)

- 1) Z is the mode set array.
- 2) ID will contain the integer ID of the display on return. Each ID assigned will be one greater than the previous.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

SUBJEG: Sets Up Subject Space

PURPOSE

Establishes the limits of the user's coordinate system (subject space), then calculates the scale factors.

USAGE

Standard Calls

CALL SUBJEG(Z,XMIN,YMIN,XMAX,YMAX)

- 1) Z is the mode set array.
- 2) XMIN is the minimum x-coordinate of the user's data.
- 3) YMIN is the minimum y-coordinate of the user's data.
- 4) XMAX is the maximum x-coordinate of the user's data.
- 5) YMAX is the maximum y-coordinate of the user's data.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. XMIN, YMIN, XMAX, and YMAX specify the boundaries of the user's data. XMIN and YMIN are not necessarily less than XMAX or YMAX. However, XMIN must not equal XMAX, or YMIN equal YMAX. SUBJEG recomputes the page margins so their relative positions in the new subject space are the same as in the old.

OBJCTG: Sets Up Object Space

PURPOSE

Establishes the size of the display scope (object space), then calculates the scale factors.

USAGE

Standard Calls

CALL OBJCTG(Z,XMIN,YMIN,XMAX,YMAX)

- 1) Z is the mode set array.
- 2) XMIN is the minimum x-coordinate of the screen in normalized object space.
- 3) YMIN is the minimum y-coordinate of the screen in normalized object space.
- 4) XMAX is the maximum x-coordinate of the screen in normalized object space.
- 5) YMAX is the maximum y-coordinate of the screen in normalized object space.

Special Calls

Two functions are provided to convert subject space coordinates into normalized object space.

- 1) To convert x-axis subject space to normalized object space, use the following function:
x normalized object space = XNORMZ(Z,X)
- 2) To convert y-axis subject space to normalized object space, use the following function:
y normalized object space = YNORMZ(Z,Y)

Modifications

Normalized object space is based upon a maximum value of 1.0 in the x- and y-axes. The user may change these values by making a mode set call. The new normalized object space coordinates must be used wherever normalized object is required. The mode set calls are as follows:

- 1) To change the maximum x-axis normalized object space,

CALL SETSMG(Z,19,XMAX)

XMAX is the new normalized x-axis object space. XMAX = 1.0 is the default value.

- 2) To change the maximum y-axis normalized object space,

CALL SETSMG(Z,20,YMAX)

YMAX is the new normalized y-axis object space. YMAX = 1.0 is the default value.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. XMAX and YMAX must be greater than XMIN and YMIN, respectively.

SETSMG: Sets Mode Values

PURPOSE

Sets a single value in the mode set array. Mode sets stay in effect until specifically reset, or until the system is re-initialized. All mode sets have a unique default value set by the initialization call to MODESG and by calls to RSETMG.

USAGE

Standard Calls

CALL SETSMG(Z,NO,VALUE)

- 1) Z is the mode set array.
- 2) NO is a number identifying the particular mode set.
- 3) VALUE is the mode set value. See Appendix H (p. 127) for a list of the individual mode sets possible.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

GETSMG: Retrieves Mode Values

PURPOSE

Retrieves a single value from the mode set array.

USAGE

Standard Calls

CALL GETSMG(Z,NO,VALUE)

- 1) Z is the mode set array.
- 2) NO is a number identifying the particular mode set.
- 3) VALUE will contain the mode set value on return. See Appendix H (pp. 127-130) for a description of the individual mode sets possible.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

RSETMG: Resets all Default Values in Mode Set Array

PURPOSE

Resets all values in the mode set array to their proper default values.

USAGE

Standard Calls

CALL RSETMG(Z)

Z is the mode set array.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

EXITG: Closes Down the Graphic Devices

PURPOSE

Turns off the graphic devices and closes the output files.

USAGE

Standard Calls

CALL EXITG(Z)

Z is the mode set array.

Special Calls

A call is provided to terminate the user's program abnormally without knowing the user's mode set array. The call is the same as that above, minus the reference to the mode set array:

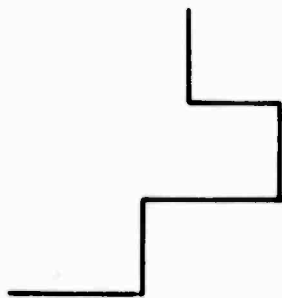
CALL EXITG(0)

MISCELLANEOUS INFORMATION

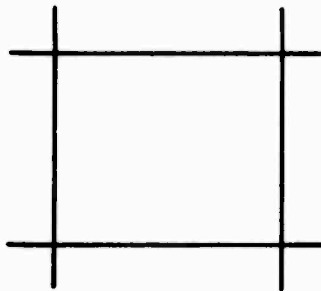
Written in FORTRAN IV.

GRAPHIC OUTPUT SUBROUTINES

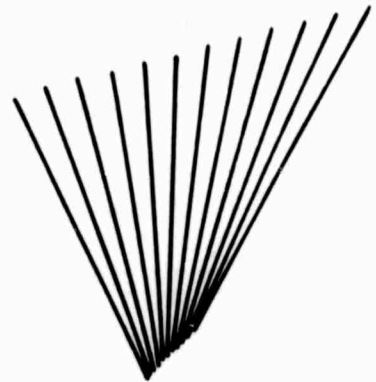
Figure 3 illustrates the graphic output subroutines.



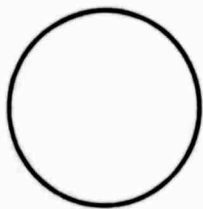
Draws joined lines--
LINESG



Draws line segments--
SEGMTG



Draws multiple lines--
MLTPLG



Draws circles and arcs--CIRARG



Plots points--POINTG

A
BC
DEFG
H*/K

Types characters--LEGNDG,TEXTG

27
36.293
0.263E + 06

Displays numeric data--NUMBRG

Fig. 3--Graphic Output Subroutines

CIRARG: Draws a Circle or an Arc

PURPOSE

Draws a circle or arc. Since the 2250 can not draw a curved line, the circle or arc is approximated by drawing short line segments.

USAGE

Standard Calls

CALL CIRARG(Z,XC,YC,RADIUS,START,ARC)

- 1) Z is the mode set array.
- 2) XC, YC is the x,y location of the center point of the arc.
- 3) RADIUS is the radius of curvature. It should be in the same units as the x coordinates.
- 4) START is the angle, in degrees, at which to start drawing. Counterclockwise angles are positive (0° is horizontal and to the right).
- 5) ARC is the number of degrees of arc to display. For example, an arc of 720° would cause two complete circles to be drawn, one on top of the other.

Modifications

Use mode sets to modify CIRARG as follows:

- 1) To change the angle between two successive points,

CALL SETSMG(Z,83,ANGLE)

ANGLE is the angle in degrees between two points on the arc. ANGLE = 5.0° is the default value.

- 2) To draw the arc with points rather than line segments,

```
CALL SETSMG(Z,82,1.)
```

To reset to draw the arc with line segments,

```
CALL SETSMG(Z,82,0.)
```

When an arc is drawn with points, all the mode sets described in POINTG are in effect. Points will be plotted at each end of an arc. Thus, if a 360° arc is specified, points will be displayed at 0° and at 360° , which turn out to be the same location.

- 3) To specify unprotected x,y coordinates,

```
CALL SETSMG(Z,92,1.)
```

Unprotected x,y coordinates may be manipulated individually by GPXYG.

To return the mode set to its default value of protected x,y coordinates,

```
CALL SETSMG(Z,92,0.)
```

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. All mode sets described in LINESG will be in effect if the circle is drawn with line segments, and all mode sets described in POINTG will be in effect if the circle is drawn with points.

LEGNDG: Types Characters

PURPOSE

Types characters, starting at a given x,y location. If the characters extend beyond the right margin, the line is ejected to the left margin; if beyond the bottom margin, the line is ejected to the top.

USAGE

Standard Calls

CALL LEGNDG(Z,X,Y,N,CHAR)

- 1) Z is the mode set array.
- 2) X,Y is the location of the center of the first character.
- 3) N is the number of characters to type ($N \geq 1$).
- 4) CHAR is the name of a variable or an array that contains the characters to type.

Modifications

Use mode sets to modify LEGNDG as follows:

- 1) To change the character size,

CALL SETSMG(Z,45,SIZE)

SIZE specifies the size of the characters relative to normal size. 1.0 and 1.5 are allowed for the 2250. SIZE = 1.0 is the default value.

- 2) To specify the number of lines to space up or down if the characters exceed the right margin, causing the line to be ejected to the left margin,

CALL SETSMG(Z,49,+SPACE)

SPACE specifies the number of lines to space up or down. This is equivalent to setting a typewriter carriage return for the number of lines to space. A minus spacing causes the line to be spaced down the page; and a positive spacing, up. SPACE = -1.0 is the default value.

- 3) To change the line orientation,

CALL SETSMG(Z,50,ANGLE)

ANGLE designates the angle in degrees at which the line of characters is to be displayed (0° is horizontal and to the right). Counterclockwise angles are positive. Only 0° and $+90^{\circ}$ are allowed for the 2250. This call does not change the character orientation. ANGLE = 0.0 is the default value.

- 4) To specify the margins of the page,

CALL SETSMG(Z,60,XLEFT)

XLEFT designates the left margin of the page in subject space coordinates. XLEFT = 0.0 is the default value.

CALL SETSMG(Z,61,RIGHT)

RIGHT specifies the right margin of the page in subject space coordinates. RIGHT = 4095.0 is the default value.

CALL SETSMG(Z,62,BOTTOM)

BOTTOM stipulates the bottom margin of the page in subject space coordinates. BOTTOM = 0.0 is the default value.

CALL SETSMG(Z,63,TOP)

TOP fixes the top margin of the page in subject space coordinates. TOP = 4095.0 is the default value.

- 5) To specify unprotected characters,

CALL SETSMG(Z,92,1.)

The user may write over unprotected characters with either the keyboard or RAND Tablet.

To reset to the default condition of protected characters,

CALL SETSMG(Z,92,0.)

- 6) To specify the form of the x,y coordinates used in the call,

CALL SETSMG(Z,14,TYPE)

TYPE fixes the form of the x,y coordinates.

If TYPE = 0.0, x,y are real subject space units.

= 1.0, x,y are real absolute raster units.

= 2.0, x,y are integer absolute raster units.

= 3.0, x,y are real normalized object space units.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

LINESG: Draws Joined Lines

PURPOSE

Draws connected line segments, joining points in an x,y array.

USAGE

Standard Calls

CALL LINESG(Z,N,X,Y)

- 1) Z is the mode set array.
- 2) N is the number of x,y coordinates. N - 1 joined line segments will be drawn.
- 3) X,Y are arrays of x,y coordinates. A line is drawn between each point in the arrays.

Special Calls

- 1) To draw a line from the current point position to an x,y location,

CALL LINESG(Z,1,X,Y)

- a) Z is the mode set array.
- b) X,Y is the x,y position to which to draw the line.

- 2) To position the current point to an x,y location without displaying anything,

CALL LINESG(Z,0,X,Y)

- a) Z is the mode set array.
- b) X,Y is the x,y point at which to position the beam.

Modifications

Use mode sets to modify LINESG as follows:

- 1) To specify unprotected x,y coordinates,

CALL SETSMG(Z,92,1.)

Use GPXYG to manipulate unprotected x,y coordinates.

To reset to the default condition of protected x,y coordinates,

CALL SETSMG(Z,92,0.)

- 2) To specify the form of the x,y coordinates used in the call,

CALL SETSMG(Z,14,TYPE)

TYPE designates the form of the x,y coordinates.

If TYPE = 0.0, x,y are real subject space units.

= 1.0, x,y are real absolute raster units.

= 2.0, x,y are integer absolute raster units.

= 3.0, x,y are real normalized object space units.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

MLTPLG: Draws Multiple Line Segments

PURPOSE

Draws multiple line segments. First, two line segments are drawn. Next, a given number of line segments are drawn between the two lines, in a direction similar to that of the original lines. MLTPLG facilitates drawing grid lines, crosshatching, and shading. It can draw lines at any orientation to each, not just parallel.

USAGE

Standard Calls

CALL MLTPLG(Z,N,X1,Y1,X2,Y2,X3,Y3,X4,Y4)

- 1) Z is the mode set array.
- 2) N specifies the number of lines to draw between the two given lines. If N = 0, only the two given lines will be drawn.
- 3) X1,Y1 and X2,Y2 are two points specifying a line.
- 4) X3,Y3 and X4,Y4 are two points specifying a line.

Modifications

All mode sets described in LINESG are applicable to this subroutine.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. MLTPLG has many uses not at first apparent. It is a convenient means of drawing any two line segments with a single call. If the delimiting

line segments are very short, the vectors drawn will appear as points. This allows the user to plot many points with a single call--exactly what one needs for shading. Moreover, the two delimiting lines need not be parallel.

NUMBRG: Displays Numeric Data

PURPOSE

Displays integer or real numbers.

USAGE

Standard Calls

- 1) To display an integer number in I format,

CALL NUMBRG(Z,X,Y,IFMT,INTG)

- a) Z is the mode set array.
- b) X,Y is the x,y position at which to display the number, i.e., the center of the leftmost character.
- c) IFMT is an integer indicating the number of digits to display.
- d) INTG is the number to display.

- 2) To display a real number in F format,

CALL NUMBRG(Z,X,Y,FMT,REAL)

- a) Z is the mode set array.
- b) X,Y is the x,y position at which to display the number, i.e., the center of the leftmost character.
- c) FMT is a real number that specifies the format. It is of the form w.d, where w specifies the field width, and d the number of decimal places ($d < 10$).
- d) REAL is the number to display.

- 3) To display a real number in E format,

CALL NUMBRG(Z,X,Y,-FMT,REAL)

- a) Z is the mode set array.
- b) X,Y is the x,y position at which to display the number, i.e., the center of the leftmost character.
- c) -FMT is a real number that specifies the format. It is of the form w.d, where w is the field width and d the number of decimal places. Four places must be provided for the exponent and two places for the sign and digit to the left of the decimal place. In general, $w = d + 7$.
- d) REAL is the number to display.

4) To display characters in A format (this call is equivalent to a call to LEGNDG),

CALL NUMBRG(Z,X,Y,-IFMT,CHARS)

- a) Z is the mode set array.
- b) X,Y is the x,y position at which to display the number, i.e., the center of the leftmost character.
- c) -IFMT is an integer indicating the number of characters to display.
- d) CHARS is the name of a variable or array that contains the character string to display.

Special Calls

A special call allows the format to be specified by mode sets:

CALL NUMBRG(Z,X,Y,0,VALUE)

VALUE is the name of the variable to display. The format must be specified by the following mode set calls prior to the call to NUMBRG:

1) CALL SETSMG(Z,77,WIDTH)

WIDTH is the field width.

2) CALL SETSMG(Z,78,DEC)

DEC is the number of decimal places.

3) CALL SETSMG(Z,79,FMT)

FMT is the format in which to display the number.

If FMT = 1.0, I format.

= 2.0, F format.

= 3.0, E format.

= 4.0, A format.

Modifications

All mode sets described in LEGNDG are also applicable to this subroutine.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

POINTG: Plots Points

PURPOSE

Plots arrays of points.

USAGE

Standard Calls

CALL POINTG(Z,N,X,Y)

- 1) Z is the mode set array.
- 2) N is the number of points ($N \geq 1$).
- 3) X,Y are arrays of N x,y coordinate pairs to plot.

Modifications

Use mode sets to modify POINTG as follows:

- 1) To specify a new plotting symbol (the default character is a normal-sized point),

CALL SETSMG(Z,84,VALUE)

VALUE must contain the character, left-justified, to use for plotting.

To reset to the default value of a point,

CALL SETSMG(Z,84,0.)

Plotting with the default point requires four bytes of 2250 buffer per point; plotting other symbols, ten bytes per point.

- 2) To specify unprotected x,y coordinates,

CALL SETSMG(Z,92,1.)

Use GPXYG to manipulate unprotected x,y coordinates.

To reset to the default condition of protected
x,y coordinates,

CALL SETSMG(Z,92,0.)

- 3) To specify the form of the x,y coordinates used
in the call,

CALL SETSMG(Z,14,TYPE)

TYPE designates the form of the x,y
coordinates.

If TYPE = 0.0, x,y are real subject space
units.

= 1.0, x,y are real absolute raster
units.

= 2.0, x,y are integer absolute
raster units.

= 3.0, x,y are real normalized
object space units.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

SEGMTG: Draws Line Segments

PURPOSE

Draws noncontiguous line segments. Arrays X1,Y1 specify the initial x,y coordinates of each segment; arrays X2,Y2 specify the terminal x,y coordinates.

USAGE

Standard Calls

CALL SEGMTG(Z,N,X1,Y1,X2,Y2)

- 1) Z is the mode set array.
- 2) N is the number of line segments to draw. If $N > 1$, X1,Y1 and X2,Y2 must be arrays of points.
- 3) X1,Y1 are arrays containing the starting points of each line.
- 4) X2,Y2 are arrays containing the terminal points of each line.

Modifications

All mode sets described in LINESG are also applicable to this subroutine.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

TEXTG: Types Characters

PURPOSE

Types characters, beginning at the current point position. If the characters extend beyond the right margin, the line is ejected to the left margin. TEXTG resembles LEGNDG, but begins typing where the previous display ended, while LEGNDG begins typing at a specified x,y coordinate.

USAGE

Standard Calls

CALL TEXTG(Z,N,CHAR)

- 1) Z is the mode set array.
- 2) N is the number of characters to type ($N \geq 1$).
- 3) CHAR is the name of the variable or array that contains the characters to type.

Modifications

All mode sets described in LEGNDG are also applicable to this subroutine.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. LINESG may be used to position the beam to any specified location.

GRAPH SUBROUTINES

A graph is composed of three separate parts: a grid of lines, labels of the major axis, and titles. Three corresponding subroutines produce the graphs. The user may then plot data on the graphs with the graphic output subroutines such as POINTG or LINESG.

A special subroutine, SETUPG, aids the user in computing arguments that produce aesthetically pleasing graphs. These subroutines are combined in subroutine GRAPHG to draw, label, title, and plot an entire graph.

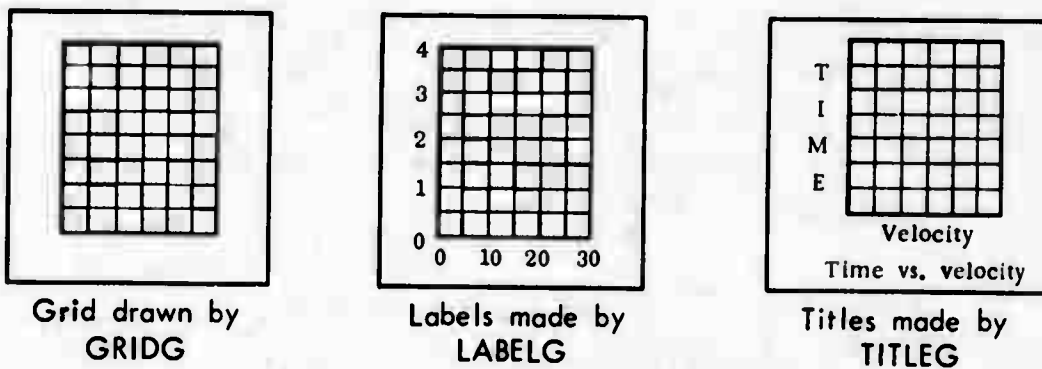


Fig. 4--Graph Subroutines

GRIDG: Draws a Grid

PURPOSE

Constructs cartesian grids--draws all lines and emphasizes the major grid divisions. GRIDG assumes that the user has established his subject space with SUBJEG and his object space with OBJCTG. The object space should provide a margin because the grid will fill the entire object space. Figure 5 illustrates the grid.

USAGE

Standard Calls

CALL GRIDG(Z,DX,DY,IXTH,JYTH)

- 1) Z is the mode set array.
- 2) DX,DY are the x,y grid intervals in user coordinates. If DX or DY = 0.0, no grid lines are drawn for the axis. DX controls spacing between the vertical grid lines and DY the horizontal. If the user's x-axis subject space decreases from left to right, or the y-axis subject space decreases from bottom to top, DX or DY must be negative, respectively. If the zero-axis lies within the grid, DX or DY are spaced from the zero-axis outward. Otherwise, they are spaced from left to right or from bottom to top.
- 3) IXTH,JYTH specify the i^{th} x-grid line and the j^{th} y-grid line to emphasize. If IXTH or JYTH = 0.0, no lines are emphasized for it. The emphasis lines are drawn with twice the normal line weight.

CALL GRIDG(Z,DX,DY,IXTH,JYTH)

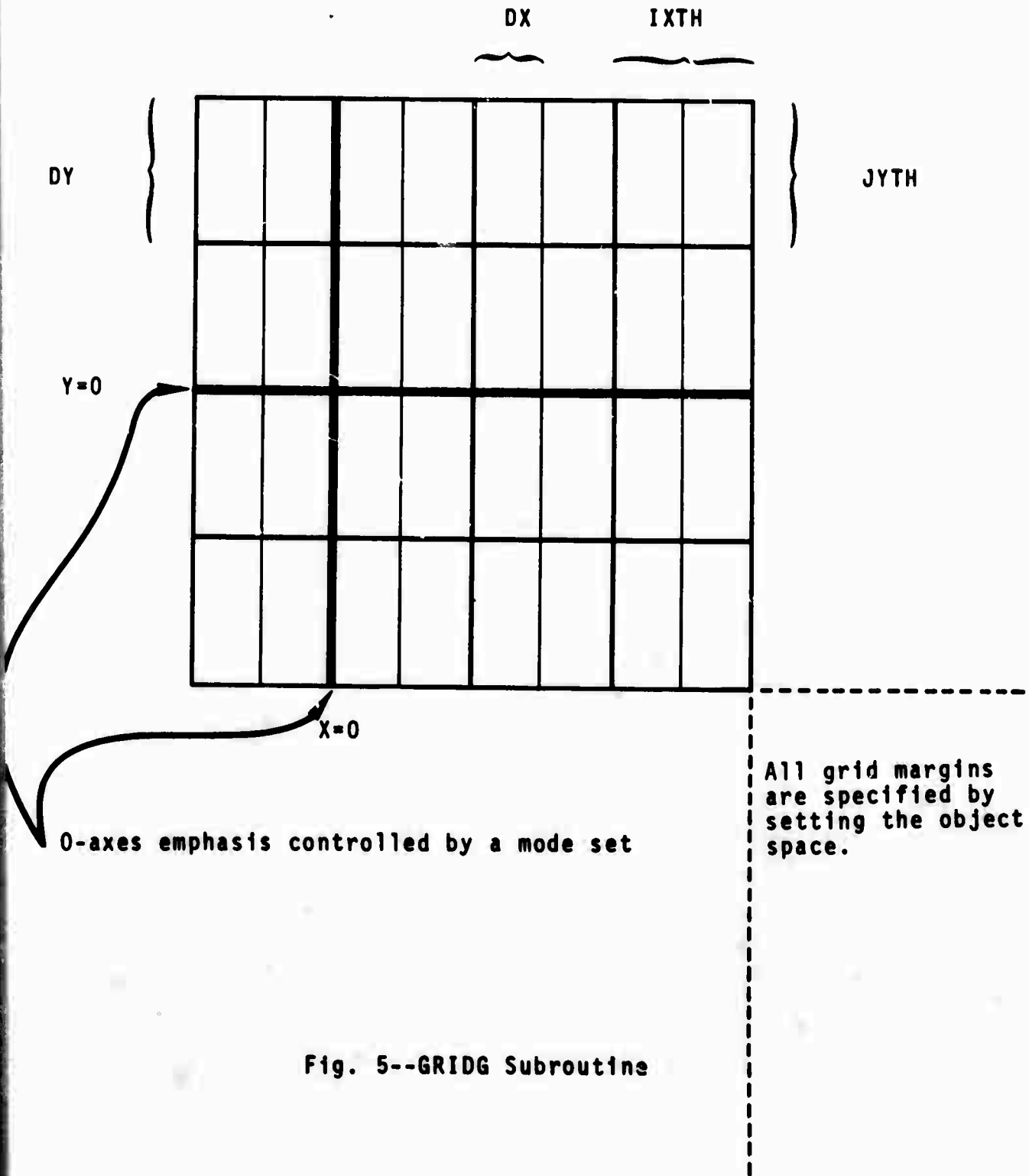


Fig. 5--GRIDG Subroutine

Special Calls

A log grid can be drawn by making one of the mode sets described below. The subject space must be positive and should be a power of ten. To draw a log grid, make the indicated mode set call and then call GRIDG. The arguments have the following special meanings for the log grids:

- 1) DX,DY are ignored and may be dummy arguments if the grid is log in x or y.
- 2) IXTH,JYTH specify whether the major cycle lines are to be emphasized. If zero, no emphasis occurs; if greater than zero, emphasis occurs.

Modifications

Use mode sets to modify GRIDG as follows:

- 1) To control the emphasis of the major axes,

CALL SETSMG(Z,100,EMPH)

EMPH specifies what is to be emphasized.

EMPH = 0.0 is the default value.

If EMPH = 0.0, emphasize $x = 0$, $y = 0$ axes.

= 1.0, emphasize $x = 0$ axis only.

= 2.0, emphasize $y = 0$ axis only.

= 3.0, no emphasis.

The emphasis lines are drawn with twice the normal line weight. They will not be drawn if the zero-axis lies outside the grid. For nonlinear grids, the major axes are defined as the $X = 1$, $Y = 1$ axes.

- 2) To draw a log grid in the x-axis,

CALL SETSMG(Z,23,1.)

To return to the default condition of a linear grid,

CALL SETSMG(Z,23,0.)

Set up the subject space for the log grid before making this mode set call.

- 3) To draw a log grid in the y-axis,

CALL SETSMG(Z,24,1.)

To return to the default condition of a linear grid,

CALL SETSMG(Z,24,0.)

Set up the subject space for the log grid before making this mode set call.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

LABELG: Labels a Grid

PURPOSE

Labels the axes of a grid drawn by GRIDG. The user may label both the x and y axes with numeric or alphanumeric labels. The y-axis will be labeled to the left of the grid and the x-axis below it. Figure 6 illustrates the labels.

USAGE

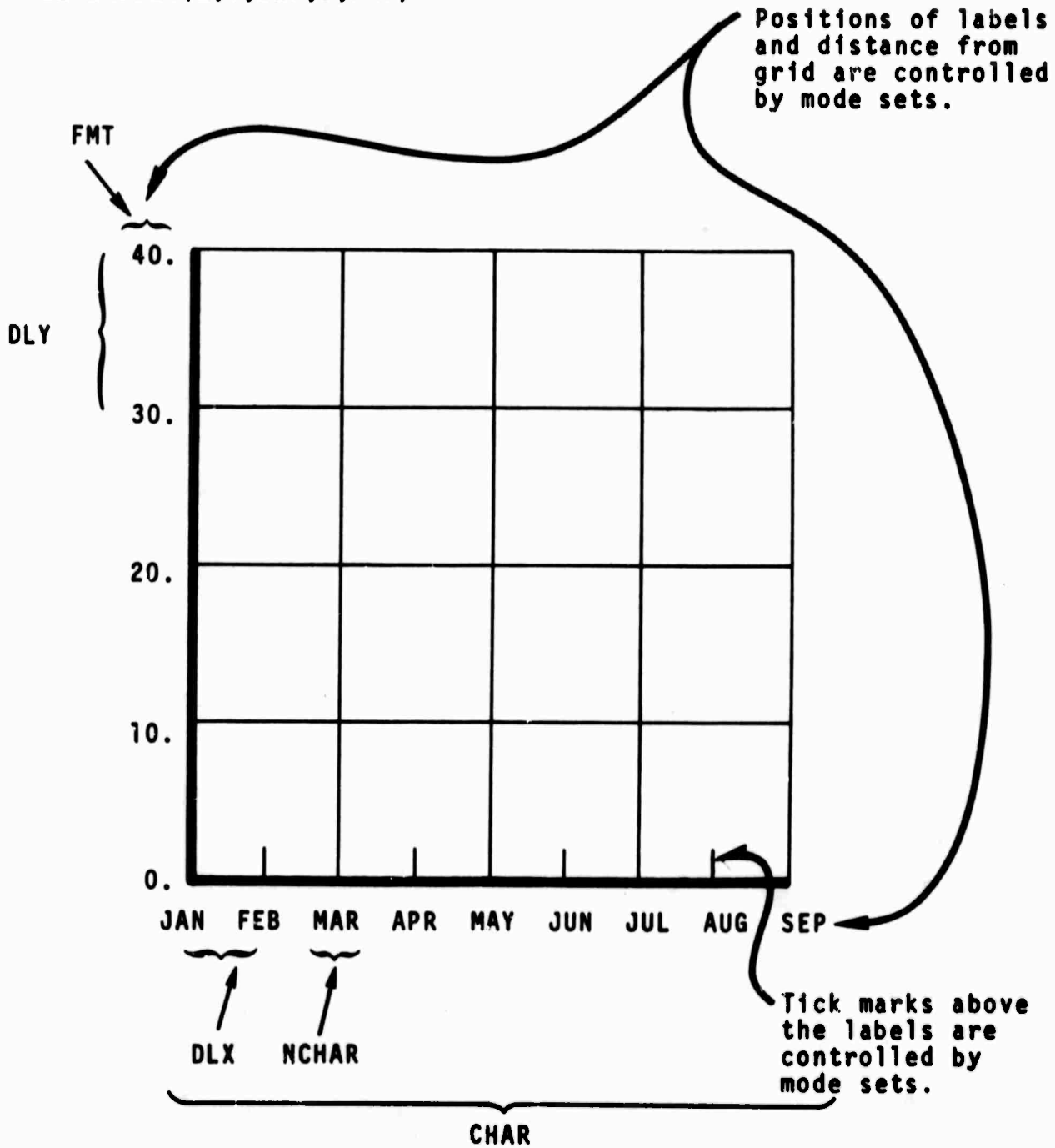
Standard Calls

- 1) To create alphanumeric labels,

CALL LABELG(Z,IAXIS,DLXY,NCHAR,CHARS)

- a) Z is the mode set array.
- b) IAXIS specifies the axis to label.
If IAXIS = 0, label the x-axis.
= 1, label the y-axis.
- c) DLXY fixes the x-interval if IAXIS = 0, or the y-interval if IAXIS = 1, in user coordinates (subject space) between each label. If the x-axis decreases from left to right, or the y-axis from bottom to top, then DLXY must be negative. For labeling log grids, DLXY specifies the cycle lines to label. For example, DLXY = 1.0 stipulates the labeling of each cycle line and DLXY = 3.0 every third line.
- d) NCHAR specifies the number of characters to display for each cycle of the label.
- e) CHAR contains the characters used in the label. CHAR must contain enough characters for all labels. For example, if NCHAR = 3

Y-axis label
CALL LABELG(Z,1,DLY,0,FMT)



X-axis label
CALL LABELG(Z,0,DLX,NCHAR,CHAR)

Fig. 6--LABELG Subroutine

and four lines are to be labeled, CHAR must contain twelve characters.

2) To create integer numeric labels in I format,

CALL LABELG(Z,IAXIS,DLXY,0,IFMT)

a) Z is the mode set array.

b) IAXIS specifies the axis to label.

If IAXIS = 0, label the x-axis.

= 1, label the y-axis.

c) DLXY fixes the x-interval if IAXIS = 0, or the y-interval if IAXIS = 1, in user coordinates (subject space) between each label. If the x-axis decreases from left to right, or the y-axis from bottom to top, then DLXY must be negative. For labeling log grids, DLXY specifies the cycle lines to label. For example, DLXY = 1.0 stipulates the labeling of each cycle line and DLXY = 3.0 every third line.

d) IFMT is an integer fixing the number of digits to display.

3) To create real numeric labels in F format,

CALL LABELG(Z,IAXIS,DLXY,0,FMT)

a) Z is the mode set array.

b) IAXIS specifies the axis to label.

If IAXIS = 0, label the x-axis.

= 1, label the y-axis.

c) DLXY fixes the x-interval if IAXIS = 0, or the y-interval if IAXIS = 1, in user coordinates (subject space) between each label. If the x-axis decreases from left to right,

or the y-axis from bottom to top, then DLXY must be negative. For labeling log grids, DLXY specifies the cycle lines to label. For example, DLXY = 1.0 stipulates the labeling of each cycle line and DLXY = 3.0 every third line.

- d) FMT is a real number that specifies the format. It is of the form w.d, where w specifies the field width and d the number of decimal places ($d < 10$).

4) To create real numeric labels in E format,

CALL LABELG(Z,IAXIS,DLXY,0,-FMT)

- a) Z is the mode set array.
- b) IAXIS specifies the axis to label.
 - If IAXIS = 0, label the x-axis.
 - = 1, label the y-axis.
- c) DLXY fixes the x-interval if IAXIS = 0, or the y-interval if IAXIS = 1, in user coordinates (subject space) between each label. If the x-axis decreases from left to right, or the y-axis from bottom to top, then DLXY must be negative. For labeling log grids, DLXY specifies the cycle lines to label. For example, DLXY = 1.0 stipulates the labeling of each cycle line and DLXY = 3.0 every third line.
- d) -FMT is a real number that specifies the format. It is of the form w.d, where w is the field width and d the number of decimal places. Four places must be provided for the exponent and two for the sign and digit to the left of the decimal place. In general, $w = d + 7$.

Modifications

Use mode sets to modify LABELG as follows:

- 1) To specify that a tick mark be drawn above the x-axis labels or beside the y-axis labels,

CALL SETSMG(Z,102,XSIZE)

XSIZE specifies the length of the tick mark for the x-axis label in normalized object space. If XSIZE = 0.0, no tick mark is drawn. XSIZE = 0.0 is the default value.

CALL SETSMG(Z,103,YSIZE)

YSIZE fixes the length of the tick mark for the y-axis label in normalized object space. If YSIZE = 0.0, no tick mark is drawn. YSIZE = 0.0 is the default value.

All modes described in LINESG are in effect when the tick marks are drawn. If XSIZE or YSIZE are positive, the tick marks are drawn inside the grid; if minus, outside. To draw tick marks alone without a label, set NCHAR = 1, and define CHARS to contain as many Hollerith blanks as there are labels.

- 2) To specify the locations at which to label,

CALL SETSMG(Z,104,XPLACE)

XPLACE defines the distance from the x-axis to the label as a multiple of character height. XPLACE = -1.0 is the default value.

If XPLACE = 0.0, label at y = 0 major axis. (This may cause the label to fall inside the grid.)

> 0.0 (+), label the x-axis
above the grid.

< 0.0 (-), label the x-axis
below the grid.

CALL SETSMG(Z,105,YPLACE)

YPLACE defines the distance from the y-axis
to the label as a multiple of character width.
YPLACE = -1.0 is the default value.

If YPLACE = 0.0, label at x = 0 major axis.

(This may cause the label
to fall inside the grid.)

> 0.0 (+), label the y-axis on
the right of the grid.

< 0.0 (-), label the y-axis on
the left of the grid.

- 3) To change the character size, give the mode set
calls described in LEGNDG.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

TITLEG: Titles a Grid

PURPOSE

Titles the x and y axes and the graph, and automatically centers the titles. TITLEG provides a quick, easy method of titling a graph. For full control over titling, use LEGNDG rather than TITLEG. Figure 7 illustrates the titles.

USAGE

Standard Calls

CALL TITLEG(Z,NX,XCHAR,NY,YCHAR,NT,TCHAR)

- 1) Z is the mode set array.
- 2) NX specifies the number of characters in the x-axis title, centered below the grid. If NX = 0, the x-axis is not titled.
- 3) XCHAR contains the NX characters for the x-axis title. If NX = 0, XCHAR should be a dummy argument.
- 4) NY specifies the number of characters in the y-axis title, vertically centered left of the grid. The line orientation will be 90°. If NY = 0, the y-axis is not titled.
- 5) YCHAR contains the NY characters for the y-axis title. If NY = 0, YCHAR should be a dummy argument.
- 6) NT specifies the number of characters in the graph title, centered below the x-axis title. If NT = 0, the graph is not titled.
- 7) TCHAR contains the NT characters for the graph title. If NT = 0, TCHAR should be a dummy argument.

CALL TITLEG(Z,NX,XCHAR,NY,YCHAR,NT,TCHAR)

Area available for titling is set in mode set array as two points by LABELG. If LABELG has not been called prior to the call to TITLEG, the user must set the values in the mode set array.

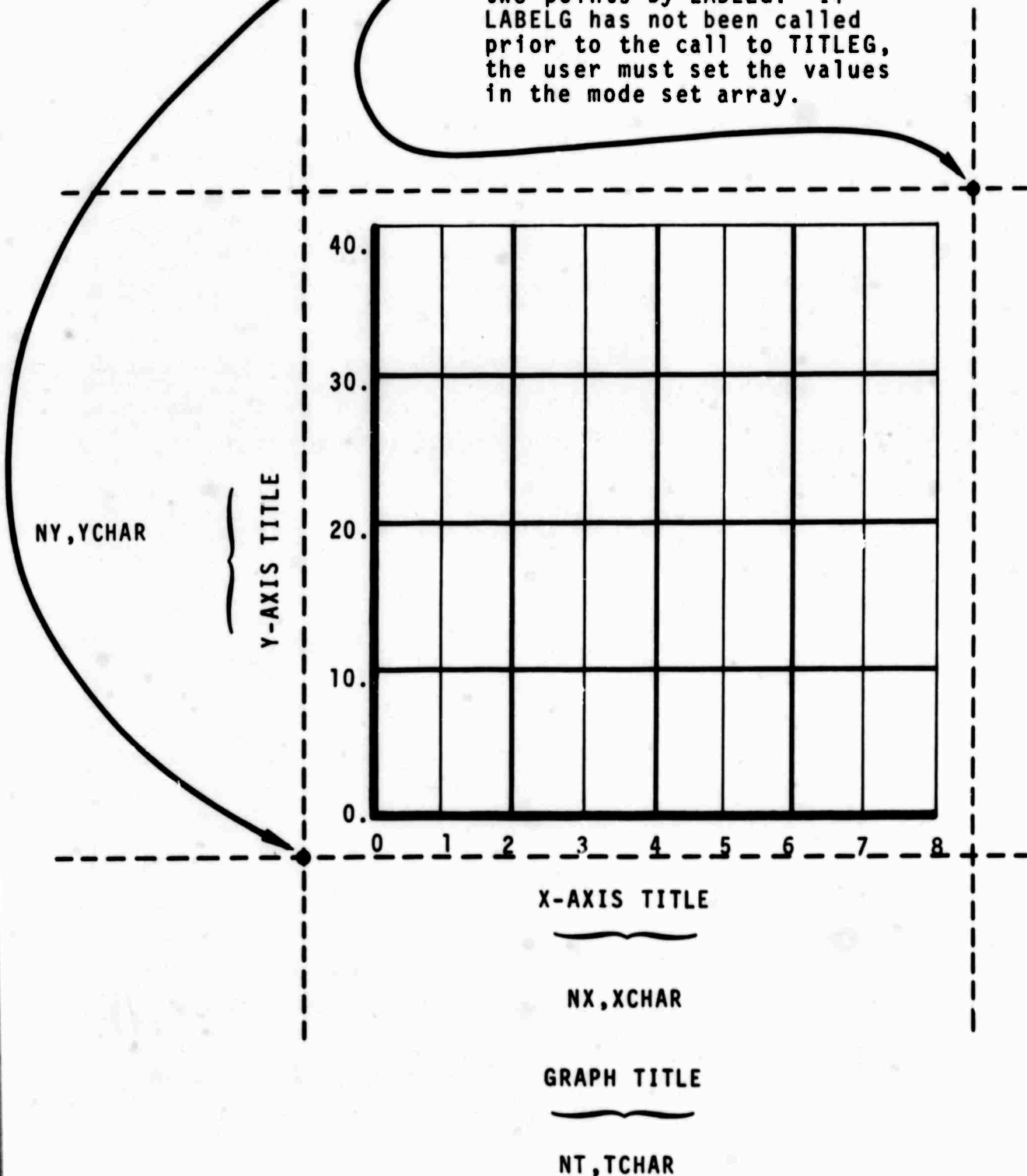


Fig. 7--TITLEG Subroutine

Modifications

Use mode sets to modify TITLEG as follows:

- 1) The mode set array contains the area available for titling. Normally, these values are stored in the mode set array by GRIDG or LABELG. However, if neither GRIDG nor LABELG has been called, the user must supply the values himself with one of the following mode set calls:

CALL SETSMG(Z,113,XMIN)

XMIN specifies the minimum x-location in normalized object space occupied by the graph, including labels.

CALL SETSMG(Z,114,YMIN)

YMIN designates the minimum y-location in normalized object space occupied by the graph, including labels.

CALL SETSMG(Z,115,XMAX)

XMAX specifies the maximum x-location in normalized object space occupied by the graph, including labels.

CALL SETSMG(Z,116,YMAX)

YMAX stipulates the maximum y-location in normalized object space occupied by the graph, including labels.

- 2) To specify the position of the title,

CALL SETSMG(Z,104,XPLACE)

XPLACE designates the distance from the grid to the x-axis title as a multiple of character height. It also fixes the distance from the x-axis title to the graph title. XPLACE = -1.0 is the default value.

If XPLACE > 0.0 (+), title above the grid.
< 0.0 (-), title below the grid.

CALL SETSMG(Z,105,YPLACE)

YPLACE specifies the distance from the y-axis to the title as a multiple of character width. YPLACE = -1.0 is the default value.

If YPLACE > 0.0 (+), title on the right side of the grid.
< 0.0 (-), title on the left side of the grid.

- 3) Change the character size with the mode set calls described in LEGNDG.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

SETUPG: Computes Appropriate Arguments for
the Grid Routines

PURPOSE

Computes values to be subsequently included in calls to GRIDG and LABELG. SETUPG allows the user to adjust his subject and object space to produce a more aesthetic graph. Altering the object space reduces the size of the display surface, creating a margin around the grid. Such alteration also causes the scaling factors to be recomputed so that all subsequent calls to graphic output subroutines will result in the data being scaled into the area of the grid. If, after calling SETUPG, the user wants to draw outside the grid, he must either call OBJCTG to reset the display surface, or use the special mode set indicating absolute raster units.

In many instances, particularly where the range of the data is unknown, it is difficult to determine in advance the arguments for the grid routines. SETUPG analyzes the range of the user's data and computes appropriate arguments for the grid routines. SUBJEG must be called before SETUPG to establish the subject space so that SETUPG will know the limits of the data. However, the subject space must not be negative; the x-axis must increase from left to right, and the y-axis from bottom to top. SETUPG computes values for both linear and nonlinear grids.

USAGE

Standard Calls

CALL SETUPG(Z,MODE,DX,DY,IXTH,JYTH,DLX,DLY,XFMT,YFMT)

- 1) Z is the mode set array.
- 2) MODE indicates whether the user's subject space and the grid margins are to be adjusted.

If MODE = 0, no adjustment is necessary.
= 1, adjust both the subject and object space.
= 2, adjust only the subject space.
= 3, adjust only the object space.

The user must call SUBJEG, giving the minimum and maximum limits of his data, before calling SETUPG.

- 3) DX,DY contain the x,y grid intervals used in subsequent calls to GRIDG.
- 4) IXTH,JYTH contain numbers, used in calls to GRIDG, specifying the x,y grid lines to emphasize.
- 5) DLX,DLY contain the x and y intervals, used in calls to LABELG, between the axes labels.
- 6) XFMT,YFMT contain the numeric formats for labels, also used in calls to LABELG.

Modifications

Use mode sets to modify SETUPG as follows:

- 1) To force SETUPG to compute values that will result in a square grid,

CALL SETSMG(Z,110,1.)

To reset the mode to its default value so that a square grid is not mandatory,

CALL SETSMG(Z,110,0.)

- 2) To adjust the density of the grid lines,

CALL SETSMG(Z,111,XDEN)

XDEN specifies the density, expressed as a minimum distance between grid lines in normalized object space, of the x-axis (vertical) grid interval lines. XDEN = .01009 is the default value (about one character width).

CALL SETSMG(Z,112,YDEN)

YDEN specifies the density, expressed as a minimum distance between grid lines in normalized object space, of the y-axis (horizontal) grid interval lines. YDEN = .016607 is the default value (about one character height).

3) To reset the grid margins,

CALL SETSMG(Z,106,XLEFT)

XLEFT is the size of the left margin, in normalized object space. XLEFT = .15 is the default value.

CALL SETSMG(Z,107,RIGHT)

RIGHT is the size of the right margin, in normalized object space. RIGHT = .15 is the default value.

CALL SETSMG(Z,108,BOTTOM)

BOTTOM is the size of the bottom margin, in normalized object space. BOTTOM = .15 is the default value.

CALL SETSMG(Z,109,TOP)

TOP is the size of the top margin, in normalized object space. TOP = .15 is the default value.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

GRAPHG: Constructs a Complete Graph

PURPOSE

Constructs an entire graph, complete with grid, labels, titles, and plotted data. It requires only minimum information from the user and makes all decisions itself. GRAPHG normally resets both the subject and object space. This allows subsequent calls to POINTG or LINESG for plotting more than one graph variable. The subject space will not be reset if NO, the number of points to plot, is zero. The object space will not be reset if the grid margins are set to zero in the mode set array.

USAGE

Standard Calls

CALL GRAPHG(Z,NO,X,Y,NX,XCHAR,NY,YCHAR,NT,TCHAR)

- 1) Z is the mode set array.
- 2) NO specifies the number of x,y coordinates to plot. If NO = 0, the graph is drawn but no points are plotted. This also prevents the subject space from being reset. NO = 1 is not legal.
- 3) X,Y are arrays of NO x,y coordinates to plot. If NO = 0, X and Y should be dummy arguments.
- 4) NX specifies the number of characters in the x-axis title, centered below the grid. If NX = 0, the x-axis is not titled.
- 5) XCHAR contains the NX characters to use for the x-axis title. If NX = 0, XCHAR should be a dummy argument.

- 6) NY specifies the number of characters in the y-axis title, oriented at 90° and vertically centered to the left of the grid. If NY = 0, the y-axis is not titled.
- 7) YCHAR contains the NY characters used in the y-axis title. If NY = 0, YCHAR should be a dummy argument.
- 8) NT specifies the number of characters in the graph title, centered below the x-axis title. If NT = 0, the graph is not titled.
- 9) TCHAR contains the NT characters used in the graph title. If NT = 0, TCHAR should be a dummy argument.

Modifications

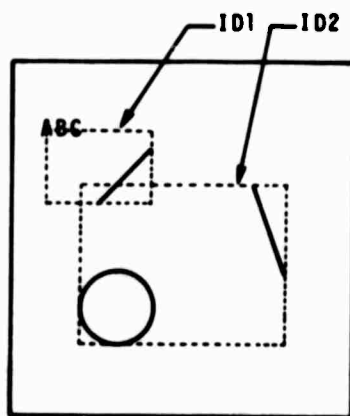
All the sets described in SETUPG, GRAPHG, LABELG, TITLEG, and POINTG are also applicable to this subroutine.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. GRAPHG resets the grid margins in the mode set array to zero because GRAPHG sets the object space based upon the current object space and the amount of grid margin specified in the mode set array. If GRAPHG did not set the grid margins to zero, subsequent calls to it would continue to decrease the size of the object space, resulting in smaller and smaller graphs.

GRAPHIC MANIPULATION SUBROUTINES

DISPLG deletes or turns displays on and off, GPCHRG manipulates unprotected characters, and GPXYG manipulates unprotected x,y coordinates.



Note that each display occupies a virtual area defined by its minimum and maximum coordinates. This virtual area is used by GSTATG and WAITSG in determining which ID is touched with the RAND Tablet pen. Where the areas overlap, the first ID created is assumed to be touched. This problem does not arise when a display is touched with the light pen since the hardware gives the system the buffer location of the light pen strike.

Fig. 8--Graphic Manipulation Subroutines

```
C THE FOLLOWING 'PROGRAM' ILLUSTRATES THE USE OF CONTROL AREAS.
C SEE THE SUBROUTINE WRITEUPS FOR A DESCRIPTION OF THE ARGUMENTS
C IN THE CALLS.
C SET FOR UNPROTECTED DATA SO WE CAN MANIPULATE IT.
  CALL SETSMG(Z,92,1.)
C GET AN ID FOR THE FIRST DISPLAY.
  CALL GETIDG(Z,ID1)
C DRAW THE DISPLAY FOR ID1.
  CALL LEGNDG(Z,X,Y,3,3HABC)
  CALL SEGMTG(Z,1,X1,Y1,X2,Y2)
C GET AN ID FOR THE NEXT DISPLAY.
  CALL GETIDG(Z,ID2)
C DRAW THE DISPLAY FOR ID2.
  CALL CIRARG(Z,X,Y,R,O.,360.)
  CALL SEGMTG(Z,1,X1,Y1,X2,Y2)
C NOW LET'S MANIPULATE THE DISPLAYS. REPLACE THE 'BC' in ID1
C WITH A 'D.'.
  CALL GPCHRG(Z,1,ID1,2,2,2HDE)
C TURN ID1 OFF SO THAT WE DON'T SEE IT.
  CALL DISPLG(Z,2,1,ID1)
C AND THEN TURN IT BACK ON SO WE CAN SEE IT AGAIN.
  CALL DISPLG(Z,1,1,ID1)
C NOW DELETE EVERYTHING.
  CALL DISPLG(Z,0,0,0)
```

DISPLG: Deletes or Turns Displays Off or On

PURPOSE

Deletes displays or turns them off or on. A display that is turned off disappears from the 2250 scope, but can be made to reappear by turning it on. A deleted display is removed entirely.

USAGE

Standard Calls

CALL DISPLG(Z,I,NO,ID)

- 1) Z is the mode set array.
- 2) I specifies the action:
 - If I = 0, delete the displays.
 - = 1, turn the displays on.
 - = 2, turn the displays off.
- 3) NO designates the number of displays.
- 4) ID contains the ID of the displays. If NO > 1, ID must be an array of size NO.

Special Calls

To process all displays,

CALL DISPLG(Z,I,Q,Q)

- 1) Z is the mode set array.
- 2) I specifies the action:
 - If I = 0, delete the displays.
 - = 1, turn the displays on.
 - = 2, turn the displays off.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

GPCHRG: Manipulates Characters in Displays

PURPOSE

Replaces or retrieves unprotected characters within a display, which are identified by the sequence in which they are displayed. For example, if two unprotected characters are displayed, then three protected characters, then a single unprotected character, the *last* unprotected character is identified as the *third* unprotected character.

USAGE

Standard Calls

- 1) To retrieve unprotected characters,

CALL GPCHRG(Z,0,ID,NTH,NO,CHAR)

- a) Z is the mode set array.
- b) ID contains the ID of the display.
- c) NTH specifies the first unprotected character to retrieve. (Retrieve NO characters, beginning with the NTH character.) If NTH = 0, NTH = 1 is assumed.
- d) NO designates the number of unprotected characters to retrieve. If NO = 0, all unprotected characters are retrieved, and NO is set to the number of unprotected characters in the display.
- e) CHAR must be an array large enough to contain the retrieved characters.

- 2) To replace unprotected characters,

CALL GPCHRG(Z,1,ID,NTH,NO,CHAR)

- a) Z is the mode set array.
- b) ID contains the ID of the display.
- c) NTH specifies the first unprotected character to replace. (Replace NO characters, beginning with the NTH character.) If NTH = 0, NTH = 1 is assumed.
- d) NO fixes the number of unprotected characters to replace.
- e) CHAR contains the characters used in replacement.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

GPXYG: Manipulates X,Y Coordinates within a Display

PURPOSE

Retrieves, replaces, or moves specific unprotected x,y coordinates within a display, which are identified by the sequence number in which they are displayed. For example, if two unprotected coordinates are displayed, then three protected coordinates, then a single unprotected coordinate, the *last* unprotected coordinate is identified as the *third* unprotected coordinate.

USAGE

Standard Calls

CALL GPXYG(Z,I,ID,NTH,NO,X,Y)

- 1) Z is the mode set array.
- 2) I indicates the manipulation to be done.
If I = 0, retrieve unprotected x,y coordinates.
= 1, replace unprotected x,y coordinates.
= 2, move unprotected x,y coordinates.
- 3) ID contains the ID of the display.
- 4) NTH specifies the unprotected x,y coordinate at which to begin manipulation. If NTH = 0, NTH = 1 is assumed.
- 5) NO designates the number of unprotected x,y coordinates to manipulate. If NO = 0, all unprotected x,y coordinates are manipulated, and NO is set to the number of unprotected x,y coordinates in the display. That is, one is telling the system to retrieve, replace, or move NO x,y coordinates, starting with the NTH coordinate.

6) X,Y are the x,y coordinates.

If I = 0, X,Y will contain the x,y coordinates on return. (X,Y must be arrays large enough to contain the coordinates.)

= 1, X,Y must be arrays of coordinates to replace the coordinates in the display.

= 2, X,Y are a single delta x and delta y to add to the original x,y coordinates.

Special Calls

A special call allows IGS to retrieve unprotected x,y coordinates, normally obtained from internal core accounting, directly from the 2250 buffer:

CALL GPXYG(Z,3,ID,NTH,NO,X,Y)

- 1) Z is the mode set array.
- 2) ID contains the ID of the display.
- 3) NTH specifies the unprotected x,y coordinate at which to begin manipulation. If NTH = 0, NTH = 1 is assumed.
- 4) NO designates the number of unprotected x,y coordinates to retrieve. If NO = 0, all unprotected x,y coordinates are retrieved, and NO is set to the number of unprotected x,y coordinates in the display.
- 5) X,Y will contain the x,y coordinates on return. (X,Y must be arrays large enough to contain the coordinates.)

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

GRAPHIC INPUT SUBROUTINES

Graphic input subroutines determine the status of graphic input devices (GSTATG) and wait for some action from graphic input devices (WAITSG). CHARG allows the user to replace unprotected characters manually from the console with the RAND Tablet pen, and DATAG allows him to input numeric data from the console. RECOG recognizes and displays geometric figures hand-drawn with the RAND Tablet pen.

GSTATG: Checks the Status of a Device

PURPOSE

Checks the current status of a graphic input device. The RAND Tablet pen is either up or down. The status of the light pen, keyboard, and function keyboard is more complicated. They are checked by both GSTATG and WAITSG. GSTATG indicates what has happened to them since the last time they were checked. Therefore, their current status may be defined as the first action since the last call to either GSTATG or WAITSG.

USAGE

Standard Calls

- 1) To check the status of the Tablet pen,

CALL GSTATG(Z,0,IVAL,X,Y)

- a) Z is the mode set array.
- b) IVAL contains the pen status.

If IVAL = 0, pen is up.

= 1, pen is down but no display was touched.

> 1, pen is down and IVAL contains the ID of the display touched.

- c) X,Y contains the current location of the pen.

- 2) To check the status of the light pen,

CALL GSTATG(Z,1,IVAL,X,Y)

- a) Z is the mode set array.

b) IVAL contains the light pen status.

If IVAL = 0, no light pen strike.

≠ 0, IVAL contains the ID of the display touched.

c) X,Y contains the x,y coordinates of the light pen strike--if there is one.

3) To check the status of the keyboard,

CALL GSTATG(Z,2,IVAL,0,0)

a) Z is the mode set array.

b) IVAL contains the keyboard status.

If IVAL = 0, no key has been hit.

= 1, END key was last hit.

= 2, CANCEL key was last hit.

4) To check the status of the function keys,

CALL GSTATG(Z,3,IVAL,IOVRL,0)

a) Z is the mode set array.

b) IVAL contains the function keyboard status. IVAL = N, where N is an integer (0 ≤ N ≤ 32).

If IVAL = 0, no key was hit.

≠ 0, IVAL is an integer indicating the last key that was struck.

c) IOVRL contains the overlay code--if a key was struck (0 ≤ IOVRL ≤ 255).

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. When the user is waiting for the Tablet or light pen, IVAL returns with the ID of any display touched. If more than one display occupies the same area, the ID of the first display created

is returned. The RAND Tablet determines the proper ID based upon the area of the display defined by two points: one point determined by the minimum, and the other by the maximum, x and y of any part of the display. Therefore, while the display may not be rectangular, the area it defines is assumed to be. Thus, some incorrect IDs may be returned when one ID overlays another and the RAND Tablet is used.

This problem does not occur with the light pen. The light pen detects the light from the CRT beam when the beam enters the pen's field of vision. When the light pen detect occurs, the location of the display in the buffer is made available. With this information, the system easily determines the ID of the display occupying that portion of the 2250 buffer.

WAITSG: Waits for Console Action

PURPOSE

Awaits user action on one of the graphic input devices.

USAGE

Standard Calls

- 1) To await a specific Tablet pen action,
CALL WAITSG(Z,0,ICHECK,IVAL,X,Y)
 - a) Z is the mode set array.
 - b) ICHECK specifies the action to await.
If ICHECK = 0, wait for pen up.
= 1, wait for pen down.
 - c) IVAL contains the ID of the display touched.
If IVAL = 0, no display touched.
> 0, IVAL contains the ID of the display.
 - d) X,Y contains the x,y coordinates of the pen on return.
- 2) To await any action on the input devices,
CALL WAITSG(Z,1,ICHECK,IVAL,X,Y)
 - a) Z is the mode set array.
 - b) ICHECK indicates, on return, which device caused the interruption.
If ICHECK = 0, Tablet pen has gone up or down.
(1) IVAL contains the pen status.

If IVAL = 0, pen up.
= 1, pen down but no display was touched.
> 1, pen down and IVAL contains the ID of the display touched.

(ii) X,Y contains the x,y coordinates of the pen on return.

If ICHECK = 1, light pen trap has occurred.

(i) IVAL contains the ID of the display touched.

(ii) X,Y contains the x,y location of the light pen strike.

If ICHECK = 2, keyboard trap has occurred.

(i) IVAL = 1, END key hit.
= 2, CANCEL key hit.

(ii) X,Y are meaningless.

If ICHECK = 3, function keyboard trap has occurred.

(i) IVAL indicates the specific key hit ($1 \leq \text{IVAL} \leq 32$).

(ii) X contains an integer representing the overlay code ($0 \leq X \leq 255$).

(iii) Y is meaningless.

3) To await action on the light pen only,

CALL WAITSG(Z,2,ICHECK,IVAL,X,Y)

4) To await action on the keyboard only,

CALL WAITSG(Z,3,ICHECK,IVAL,X,Y)

5) To await action on the function keys only,

CALL WAITSG(Z,4,ICHECK,IVAL,X,Y)

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. When the user is awaiting action on the Tablet or light pen, IVAL returns with the ID of any display touched. If more than one display occupies the same area, the ID of the first display created is returned. The RAND Tablet determines the proper ID based upon the area of the display defined by two points: one point determined by the minimum, and the other by the maximum, x and y of any part of the display. Therefore, while the display may not be rectangular, the area it defines is assumed to be. Thus, some incorrect IDs may be returned when one ID overlays another and the RAND Tablet is used.

This problem does not occur with the light pen. The light pen detects the light from the CRT beam when the beam enters the pen's field of vision. When the light pen detect occurs, the location of the display in the buffer is made available. With this information, the system easily determines the ID of the display occupying that portion of the 2250 buffer.

CHARG: Inputs Characters with the RAND Tablet

PURPOSE

Allows the user to write over unprotected characters manually with the RAND Tablet pen. The handwritten character will replace the character on the scope, and the ID of the appropriate display will be flagged as having had a character changed. The new character will be the same size (normal or large) as the original character.

USAGE

Standard Calls

CALL CHARG(Z)

Z is the mode set array. The user may write as many characters as he desires. When finished, he may cause a return by drawing a horizontal line more than one inch long (usually accomplished by sweeping the pen across the Tablet) or by hitting the END key, a function key, or using the light pen.

Modifications

To change the pen-up delay time (time required for IGS to decide whether two pen strokes constitute a single character),


CALL SETSMG(Z,88,TIME)

TIME specifies the time in seconds to allow the user to delay in writing multi-stroke characters.
TIME = 0.5 is the default value.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. The following characters may be recognized: A through Z, 0 through 9,

and . , + - * / = \$ () ' > < . When an unrecognizable character is drawn, a question mark (?) will be returned as the character. The following characters may be recognized indirectly:

<u>Character drawn</u>	<u>Character recognized</u>
Caret ^	ε
Left bracket [
Right bracket]	┘
Scrub (erase) 	Blank

References 3 and 4 completely describe the character recognition routines. CHARG depends on the RAND Tablet.

DATAG: Inputs Data

PURPOSE

Allows the user to input data from the 2250 console. He passes it a set of variables, whose current values are subsequently displayed on the scope. He may modify the values with either the keyboard or the RAND Tablet. When he causes a return from the subroutine, the displays are erased and the new values of the variables are returned to the user. A return is accomplished by hitting the END key, a function key, using the light pen, or by drawing a horizontal line more than one inch long with the RAND Tablet pen (usually by sweeping the pen across the Tablet).

USAGE

Standard Calls

CALL DATAG(Z,X,Y,NO,NAME1,FMT1,VAR1,NAME2,FMT2,VAR2,...)

- 1) Z is the mode set array.
- 2) X,Y is the x,y location at which to display the first variable. Each successive variable is displayed below the previous.
- 3) NO specifies the number of variables to display. Three arguments are needed for each variable: a label, the format, and the variable itself ($1 \leq NO \leq 20$).
- 4) NAME must contain eight characters to display to the left of the variable on the 2250 scope. The name helps the user identify the variables being displayed.
- 5) FMT designates the format in which to display the variable. For integer (I) format, FMT must be a positive integer specifying the number of

digits. For real format, FMT must be a real number of the form w.d, where w is the field width and d is the number of decimal places ($d < 10$). If FMT is +, the output is in F format. If FMT is -, the output is in E format. For alphameric (A) format, FMT must be a negative integer specifying the number of characters.

- 6) VAR is the variable to be displayed. It may contain a real, integer, or character value on entry and is updated on return.

Modifications

Use mode sets to modify DATAG as follows:

- 1) To change the automatic spacing of the variables,
CALL SETSMG(Z,89,DX)
CALL SETSMG(Z,90,DY)
DX,DY are the distances between each successive variable in the x and y directions.
DX = 0.0, DY = -80.0 are the default values.
- 2) To specify the location of each variable rather than having it automatically calculated,
CALL SETSMG(Z,91,1.)

This call informs DATAG that the X,Y in the calling sequence are arrays of x,y coordinates of size NO. To reset the mode to its default condition where only the first x,y location of the buttons is specified,

CALL SETSMG(Z,91,0.)

- 3) To specify the number of characters in the names to be associated with the variables,
CALL SETSMG(Z,93,CN)

CN is a real number specifying the number of characters in each name. CN = 8.0 is the default value.

MISCELLANEOUS INFORMATION

Written in two parts. DATAG, an assembly language routine, collects the variable-length calling sequence into an array and calls DATAGG. DATAGG is written in FORTRAN IV and does all input/output.

The cursor is positioned on the first variable, to allow the user to type in values from the keyboard. Upon return, the cursor resumes its initial position.

Return is initiated either by hitting the END key or by drawing a horizontal line more than one inch long with the RAND Tablet pen.

RECOG: Recognizes and Displays Geometric Figures

PURPOSE

Recognizes and displays the following hand-drawn geometric figures: lines, rectangles, circles, triangles, diamonds, ellipses, and trapezoids. RECOG returns when a figure is recognized. The user may force a return by hitting a function key or the END key, or by using the light pen.

USAGE

Standard Calls

CALL RECOG(Z,DATA)

- 1) Z is the mode set array.
- 2) DATA must be an array. It will contain the following on return:

If DATA(1) = 0.0, line.
 = 1.0, rectangle.
 = 2.0, circle.
 = 3.0, triangle.
 = 4.0, ellipse.
 = 5.0, diamond.
 = 6.0, trapezoid.
 = 100.0, return caused by END key,
 function key, or light pen,
 No figure was recognized
 or drawn.

DATA(2) = x location of pen down.
DATA(3) = y location of pen down.
DATA(4) = x location of pen up.
DATA(5) = y location of pen up.
DATA(6) = x location of centroid.
DATA(7) = y location of centroid.

DATA(8) = minimum x of figure.
DATA(9) = minimum y of figure.
DATA(10) = maximum x of figure.
DATA(11) = maximum y of figure.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. RECOG depends on the RAND
Tablet.

PROGRAM CONTROL SUBROUTINES

Three program control subroutines are provided: AREASG and BUTTNG establish "sensitive" areas, and PUSHG determines when these areas are touched.

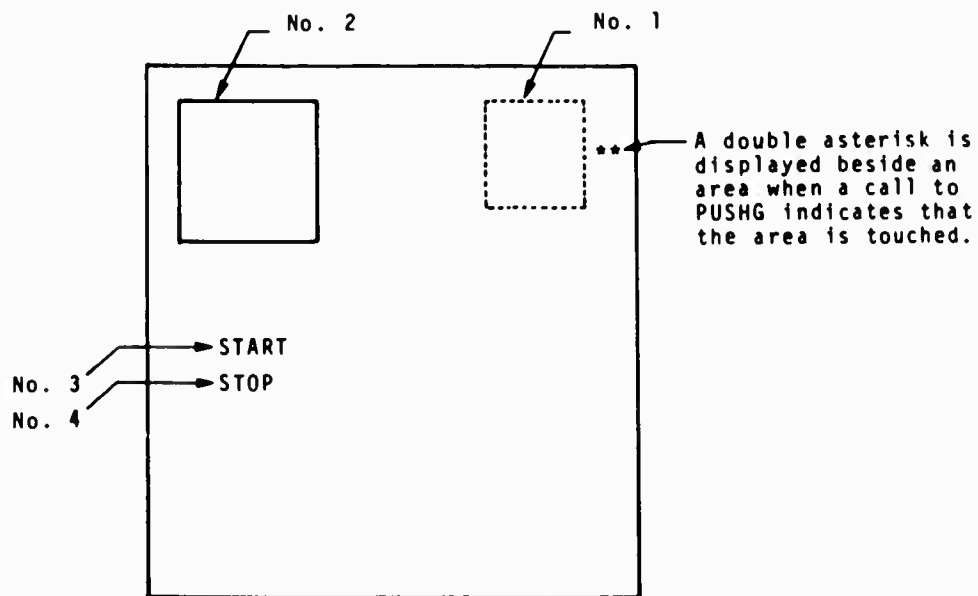


Fig. 9--Program Control Subroutines

```
C THE FOLLOWING 'PROGRAM' ILLUSTRATES THE USE OF CONTROL AREAS.
C SEE THE SUBROUTINE WRITEUPS FOR A DESCRIPTION OF THE ARGUMENTS
C IN THE CALLS.
C FIRST, CREATE A VIRTUAL AREA AND ASSIGN IT SEQUENCE NO. 1.
  CALL AREASG(Z,1,X1,Y1,X2,Y2,1)
C THEN SET THE MODE SO THAT A BOX IS DRAWN AROUND THE NEXT AREA.
  CALL SETSMG(Z,98,1.)
C THE NEXT AREA IS CREATED AND ASSIGNED SEQUENCE NO. 2.
  CALL AREASG(Z,1,X3,Y3,X4,Y4,2)
C FINALLY WE CREATE THE TWO BUTTONS AND ASSIGN THEM SEQUENCE NO'S
C 3 AND 4.
  CALL BUTTNG(Z,X,Y,2,16HSTART  STOP  ,3)
C I CAN NOW WAIT FOR SOME BUTTON OR AREA TO BE TOUCHED BY THE USER
C AT THE CONSOLE.
  CALL PUSHG(Z,0,0,ISEQ)
C AND THEN USE A COMPUTED GO TO TO PROCESS THE BUTTON.
  GO TO (100,200,300,400),ISEQ
```

AREASG: Sets Up Rectangular Control Areas
on the 2250 Scope

PURPOSE

Defines sensitive control areas on the 2250 screen. Once the rectangular control areas are set up, use PUSHG to detect when they are touched by either the RAND Tablet pen or light pen.

USAGE

Standard Calls

CALL AREASG(Z,N,X1,Y1,X2,Y2,ISEQ)

- 1) Z is the mode set array.
- 2) N is the number of control areas to set up. If $N > 1$, X1,Y1 and X2,Y2 must be arrays of at least size N.
- 3) X1,Y1 is the lower left point of the area.
- 4) X2,Y2 is the upper right point of the area.
- 5) ISEQ is the sequence number assigned to the first area. Each succeeding area is assigned a sequence number one greater than the previous area, so the final number equals $ISEQ + N - 1$. PUSHG will return this number when the control area is touched.

Modifications

- 1) To draw a box around the control areas to make them visible,
CALL SETSMG(Z,98,1.)

This mode set must be used if the area is to be detected with the light pen.

- 2) To reset the mode to its default value (no box drawn around the control area),

CALL SETSMG(Z,98,0.)

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. If one sensitive area should overlay another, the area most recently created will be detected. Sensitive areas turned off by DISPLG can not be detected by PUSHG until DISPLG turns them back on.

BUTTNG: Sets Up Control Buttons on the 2250

PURPOSE

Sets up alphameric buttons on the 2250 screen. Once the buttons are set up, use PUSHG to detect when they are touched by the RAND Tablet pen or light pen.

USAGE

Standard Calls

CALL BUTTNG(Z,X,Y,N,NAMES,ISEQ)

- 1) Z is the mode set array.
- 2) X,Y is the x,y centroid of the leftmost character of the first button. The buttons are displayed in columnar form with the first button at the x,y location, and each successive button placed below its predecessor.
- 3) N is the number of buttons to display.
- 4) NAMES is an array containing a character string for the button labels, each eight characters long. The first eight characters will be displayed as the first button, the second eight as the next button, etc.
- 5) ISEQ is the sequence number assigned to the first button. Each succeeding button is assigned a sequence number one greater than the previous, so the number of the last button equals ISEQ + N - 1. PUSHG returns this sequence number to indicate a button touch.

Modifications

Use mode sets to modify BUTTNG as follows:

- 1) To change the length of the button labels,

CALL SETSMG(Z,95,CN)

CN is a real number specifying the number of characters in a button label. CN = 8.0 is the default value.

- 2) To alter the automatic spacing of the buttons,

CALL SETSMG(Z,96,DX)

CALL SETSMG(Z,97,DY)

DX,DY are the distances between each successive button in the x and y directions. DX = 0.0, DY = -80.0 are the default values.

- 3) To specify the location of each button, rather than having it automatically calculated,

CALL SETSMG(Z,94,1.)

This call informs BUTTNG that the X,Y in the calling sequence are arrays of x,y coordinates of size N. To reset the mode to its default value where only the first x,y location of the buttons is specified,

CALL SETSMG(Z,94,0.)

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. If one sensitive area should overlay another, the area most recently created will be detected. Sensitive areas turned off by DISPLG can not be detected by PUSHG until DISPLG turns them back on. Because of hardware design in the 2250, the light pen can most successfully detect the leftmost characters of the buttons.

PUSHG: Determines Whether a Control Area is Pushed

PURPOSE

Determines whether the pen has touched a control area set up by BUTTNG or AREASG. If so, an asterisk will appear beside the area.

USAGE

Standard Calls

CALL PUSHG(Z,X,Y,ISEQ)

- 1) Z is the mode set array.
- 2) X,Y are the x,y coordinates of the pen, previously obtained from a call to GSTATG or WAITSG.
- 3) ISEQ contains the sequence number, specified by a call to BUTTNG and AREASG, of the control area panned. ISEQ = 0 if no area is panned.

Special Calls

If a previous x,y coordinate has not been obtained,

CALL PUSHG(Z,0,0,ISEQ)

Return occurs only when the light pen or RAND Tablet pen touches a button.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. If one sensitive area should overlay another, the area most recently created will be detected. Sensitive areas turned off by DISPLG can not be detected by PUSHG until DISPLG turns them back on.

MISCELLANEOUS SPECIAL-FUNCTION SUBROUTINES

ALARMG sounds the 2250 audio alarm; CURSRG locates, inserts, or removes the cursor from unprotected characters; and SETFKG sets the function-key lights. GTIMEG determines the date and time of day, and WTIMEG suspends computer operations for a specified period of time. CONVTG converts characters to numeric, and FMTSG converts numeric data to characters. CHANG identifies IDs that have had characters changed by the RAND Tablet, DEBUG displays selected variables, and SNAPG produces 2250 hardcopy.

ALARMG: Sounds the 2250 Audio Alarm

PURPOSE

Sounds the 2250 audio alarm, an unpleasant whistle, for about two seconds.

USAGE

Standard Calls

CALL ALARMG(Z)

Z is the mode set array.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

CHANG: Identifies the IDs that have had
Character Changes

PURPOSE

Identifies which displays have had characters changed by the RAND Tablet. It also identifies which IDs exist.

USAGE

Standard Calls

CALL CHANG(Z,0,ID,ICHNG)

- 1) Z is the mode set array.
- 2) ID is the ID of the display.
- 3) ICHNG indicates the changes on return.
If ICHNG = 0, no change.
= 1, characters were changed.

Special Calls

The following special call causes the system to return an ID and its changes to the user. This call also determines which IDs exist.

CALL CHANG(Z,1,ID,ICHNG)

- 1) Z is the mode set array.
- 2) ID must contain the last ID retrieved. It will contain the next ID on return. Set ID = 0 to retrieve the first ID. ID will equal zero on return when all IDs have been retrieved.
- 3) ICHNG indicates the change to the ID as above.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

CONVTG: Converts Characters to Numeric

PURPOSE

Converts a character string to a real or integer number.

USAGE

Standard Calls

CALL CONVTG(Z,IFMT,IW,IN,OUT)

- 1) Z is the mode set array.
- 2) IFMT specifies the conversion to be done.
If IFMT = 1, convert to integer.
= 2, convert to real.
= 3, convert to real if a decimal point or E is encountered; otherwise, convert to integer.
- 3) IW specifies the number of characters in the input string.
- 4) IN is the address of the character string to convert.
- 5) OUT will contain the converted integer or real number.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

CURSRG: Inserts or Removes Cursor

PURPOSE

Inserts the cursor into or removes it from unprotected characters on the 2250 scope. The cursor appears as a dash under a character and indicates where the next keyboard character will be typed in. Once displayed, it may be moved around with the keyboard.

USAGE

Standard Calls

- 1) To insert the cursor,

CALL CURSRG(Z,0,ID,IC)

- a) Z is the mode set array.
- b) ID is the ID of the display in which to insert the cursor.
- c) IC is the specific character position within the display at which to insert the cursor. (The characters must be unprotected for the cursor to be inserted.) $1 \leq IC \leq N$, where N is the number of characters in the display.

- 2) To remove the cursor,

CALL CURSRG(Z,1,ID,IC)

- a) Z is the mode set array.
- b) ID contains the ID of the display containing the cursor. (If ID = 0, the cursor is not on.)
- c) IC contains the character position upon which the cursor rests. $1 \leq I \leq N$, where N is the number of characters in the display.

3) To find the cursor,

CALL CURSRG(Z,2,ID,IC)

- a) Z is the mode set array.
- b) ID contains the ID of the display containing the cursor. (If ID = 0, the cursor is not on.)
- c) IC contains the character position upon which the cursor rests. $1 \leq I \leq N$, where N is the number of characters in the display.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

DEBUG: Debugging Aid

PURPOSE

Provides the user a simple call to display selected variables on the 2250 scope. After the variables are displayed, DEBUG waits for the user's next input action-- light pen, Tablet pen, function keys, or keyboard. DEBUG then erases the display and returns.

USAGE

Standard Calls

CALL DEBUG(Z,V1,V2,V3,V4,V5,V6)

- 1) Z is the mode set array.
- 2) V1,V2,...,V6 are the variables to display. They may be real or integer; DEBUG will determine which and display them in the proper format. If the user has fewer than six variables to display, he should pad the call with dummy variables. If he has more than six, he must make two calls.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. DEBUG is meant to be as simple to use as possible; thus, it is inflexible.

FMTSG: Converts Numeric Data to Characters

PURPOSE

Converts a number, either real or integer, into a character string.

USAGE

Standard Calls

CALL FMTSG(Z,IFMT,IW,ID,VALUE,CHARS)

- 1) Z is the mode set array.
- 2) IFMT specifies the type of number to convert.
If IFMT = 1, convert an integer number to I format.
= 2, convert a real number to F format.
= 3, convert a real number to E format.
- 3) IW stipulates the field width of the converted number.
- 4) ID designates the number of decimal places (ID = 0 if IFMT = 1).
- 5) VALUE is the number to convert. It may be either real or integer, depending upon the value of IFMT.
- 6) CHARS contains the character string upon return. CHARS must be an array large enough to contain IW characters.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV. If the number can not be converted, CHARS will contain asterisks (*) on return.

GTIMEG: Returns the Time of Day

PURPOSE

Returns the time of day.

USAGE

Standard Calls

CALL GTIMEG(Z,DAY,TIME)

- 1) Z is the mode set array.
- 2) DAY will contain the day of year in three left-justified EBCDIC characters.
- 3) TIME will contain the time of day, as a real number, in seconds.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

SETFKG: Sets the Function Keys

PURPOSE

Sets the function keys, numbered from 1 to 32. All 32 function keys and 2**7 forms overlays are normally available to the user. However, one forms overlay has been reserved for system usage. When it is in place, only keys 1 through 29 are available to the user. Keys 30 through 32 are reserved for haracopy. The system forms overlay has notch 7 intact.

USAGE

Standard Calls

CALL SETFKG(Z,N,KEYS)

- 1) Z is the mode set array.
- 2) N is the number of keys to turn on ($1 \leq N \leq 32$).
- 3) KEYS is an array containing the specific function keys to turn on. KEYS is an integer array, and must be $\geq N$. All function keys not specified in array KEYS will be turned off. Integer values of zero in the array are ignored. To turn specific keys on or off, do the following:

```
DIMENSION KEYS(32)
DO 100 I = 1, 32
100 KEYS(I) = I
```

To turn key I off, set KEY(I) = 0 and call SETFKG.
To turn key I on, set KEY(I) = I and call SETFKG.

Special Calls

- 1) To turn all function keys off,
CALL SETFKG(Z,0,0)

- 2) To use a 32-bit word as a mask for turning the function keys on (as might be done in assembly language),

CALL SETFKG(Z,0,MASK)

- a) Z is the mode set array.
- b) MASK is a full word. Any 1 bits in positions 0 through 31 of the word cause the corresponding function keys, 1 through 32, to be turned on. All others will be turned off.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

SNAPG: 2250 Hardcopy

PURPOSE

SNAPG reads in the 2250 buffer, converts the 2250 commands to S-C 4060 meta-language format, and writes them out onto an output device (normally tape or disk). The S-C 4060 then produces the hardcopy.

USAGE

Standard Calls

CALL SNAPG(DCBZZ,ICODE)

- 1) DCBZZ must be defined as an external within the calling program, using the following FORTRAN statement:

EXTERNAL DCBZZ

- 2) ICODE specifies the action to be taken.

If ICODE = 0, make a copy and advance frame.

= 1, make a copy but do not advance frame.

= 2, open output file. (Must be done before any other call.)

= 3, close output file. (Must be last call.)

= 4, set for 8 1/2 x 11 hardcopy (11 x 14 is assumed).

= 5, reset for 11 x 14 hardcopy.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. A DD card must be included to define the output data set as follows:

//GO.SC4060ZZ DD DSNAME=SCOOPS,DISP=(MOD,PASS)

This card is not required if the catalogued graphics procedures are used.

WTIMEG: Pauses for a Specified Length of Time

PURPOSE

Suspends execution of the user's program for a specified length of time by putting the computer into the WAIT state. WTIMEG is particularly useful for regulating the rate of display movement.

USAGE

Standard Calls

CALL WTIMEG(Z,SEC)

- 1) Z is the mode set array.
- 2) SEC is the time, in seconds, to pause.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

BLANK PAGE

Appendix A

SAMPLE PROGRAMS

This section contains a few simple sample programs.

SAMPLE PROGRAM 1

```
//TEST JOB (B273,C),RR750          PAL 49
// EXEC XFORTCLG
//FORT.SYSIN DD *
C SINCE THIS IS A SAMPLE PROGRAM, DON'T LOOK FOR ANY GREAT SIGNIFICANCE
C IN THE DATA THAT IS PLOTTED.
C DIMENSION THE MODE SET ARRAY.
  DIMENSION Z(200)
  DIMENSION X(100),Y(100)
C INITIALIZE IGS WITH A CALL TO MODESG.
  CALL MODESG(Z,0)
C COMPUTE SOME POINTS TO PLOT.
  DO 100 I = 1,100
    X(I) = (I-1)*10
  100 Y(I) = (I-1)*10
C THE DEFAULT PLOTTING SYMBOL IS A POINT. I DON'T WANT A POINT FOR MY
C PLOT SYMBOL SO I WILL CHANGE IT WITH A MODE SET CALL.
  CALL SETSMG(Z,84,1H*)
C NOW I WILL PLOT +'S INSTEAD OF POINTS.
C GET AN ID TO IDENTIFY THE DISPLAY.
  CALL GETIDG(Z,10)
C I WILL NOW CALL ON GRAPHG TO DRAW, LABEL, AND TITLE THE GRAPH, AND
C PLOT 1024 POINTS OF X,Y COORDINATES.
  CALL GRAPHG(Z, 100,X,Y,14,14H*TIME--IN HOURS,18,18HDISTANCE--IN MIL
  1ES,16,16H*TIME VS DISTANCE)
C COMPUTE SOME MORE POINTS TO PLOT ON THE SAME GRAPH.
  DO 200 I = 1,100
    X(I) = (100-I)*10
  200 Y(I) = (I-1)*10
C I WANT TO LABEL THE TOP AND RIGHT SIDE OF THE GRID. I MUST MAKE THE
C APPROPRIATE MODE SETS FOR THE LABEL POSITION.
  CALL SETSMG(Z,104,1.)
  CALL SETSMG(Z,105,1.)
C I WILL NOW LABEL THE X-AXIS.
  CALL LABELG(Z,0,200.,0,4)
C AND LIKEWISE FOR THE Y-AXIS.
  CALL LABELG(Z,1,100.,0,4)
C THEN TO ADD SOME TITLES.
  CALL TITLEG(Z,21,21HDISTANCE--IN FURLONGS,19,19H*TIME--IN FORTNIGHT
  1S,0,0)
C I SHOULD CHANGE THE PLOT SYMBOL SO I WON'T CONFUSE MY DATA.
  CALL SETSMG(Z,84,1HX)
C AND THEN PLOT THE DATA.
  CALL POINTG(Z,100,X,Y)
C NOW I WILL AWAIT SOME ACTION FROM THE 2250.
  CALL WAITSG(Z,1,ICHECK,IVAL,XX,YY)
C I AM DONE. I MUST CALL EXITG TO TERMINATE THE GRAPHIC OUTPUT.
  CALL EXITG(Z)
  CALL EXIT
  END
```

/*

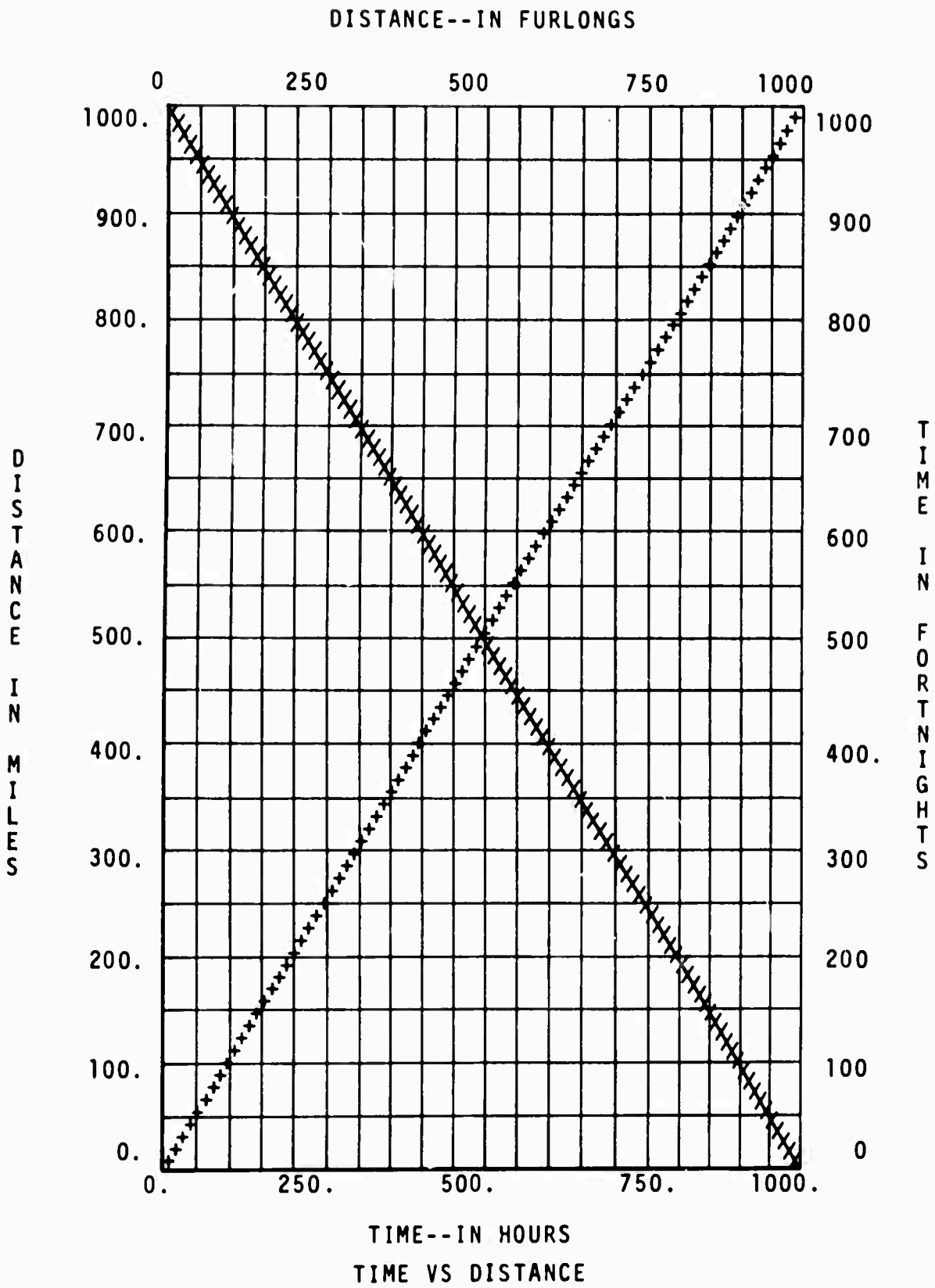


Fig. 10--Output from Sample Program 1

SAMPLE PROGRAM 2
PAL 49

```
//TEST JOB (8273,C),88750
// EXEC XFORTCLG
//FORT.SYSIN DD *
C THIS SAMPLE PROGRAM IS COPIED FROM THE "GRAPHIC PROGRAMMING SERVICES
C FOR FORTRAN IV," C27-6932-0, PAGES 85-90.
C                               BEGIN PROGRAM
C                               SET UP DIMENSION STATEMENT FOR VARIOUS ARRAYS USED.
C DIMENSION Z(200),CIRX(8),CIRY(8),LABEL(32),ID(18)
C                               INITIALIZE
C CALL MODESG(Z,0)
C
C                               DEFINE OUTER LABELS AND STORE THEM IN LABEL.
C DATA LABEL/'NUM1NUM2NUM3NUM4NUM5NUM6NUM7NUM8'/,X/'X' '/'
C                               SET LOCATION TO CENTER OF SCREEN.
C CX=2047.0
C CY=2047.0
C                               SET CHARACTER MODE FOR LARGE SIZE, PROTECTED IS
C                               ASSUMED.
C CALL SETSMG(Z,45,1.5)
C GET AN ID FOR CENTER X. ID(1) = CENTER X
C CALL GETIDG(Z,ID(1))
C DISPLAY THE CENTER X
C CALL LEGNDG(Z,CX,CY,1,X)
C                               COMPUTE COORDINATES FOR X'S OF OUTER CIRCLE.
C                               PLACE THEM IN THE CIRX AND CIRY ARRAYS FOR WHICH MAIN
C                               STORAGE HAS BEEN ALLOCATED BY DIMENSION STATEMENT.
C R=1200.0
C C=3.141596/180
C THETA= 45.0
C DO 10 I=1,8
C RAOIAN=THETA*FLOAT(I)*C
C CIRX(I)=CX+R*COS(RAOIAN)
C 10 CIRY(I)=CY+R*SIN(RAOIAN)
C                               SET CHARACTER MODE FOR BASIC SIZE.
C CALL SETSMG(Z,45,1.)
C                               GENERATE THE OUTER X'S. CALCULATE X-COORDINATE
C                               OF LABEL ASSOCIATED WITH THAT X. GENERATE LABEL AND
C                               TURN IT OFF. DO THIS 8 TIMES TO COMPLETE CIRCLE.
C DO 50 NUM=1,8
C GET AN ID FOR OUTSIDE X. ID(2) - ID(9) = OUTSIDE X.
C CALL GETIOG(Z,ID(NUM+1))
C 50 CALL LEGNDG(Z,CIRX(NUM),CIRY(NUM),1,X)
C GET AN ID FOR LABELS. ID(10) = LABEL.
C CALL GETIOG(Z,ID(10))
C DO 60 NUM=1,8
C CXNUM = CIRX(NUM) + 100.
C 60 CALL LEGNDG(Z,CXNUM,CIRY(NUM),4,LABEL(NUM))
C TURN OFF LABELS.
C CALL DISPLG(Z,2,1,ID(10))
C                               GENERATE CIRCLES AROUND OUTER X'S AND TURN THEM OFF.
C DO 110 NUM=1,8
C GET AN ID FOR THE CIRCLES. ID(11) - ID(18) = CIRCLE.
C CALL GETIDG(Z,ID(NUM+10))
C 110 CALL CIRARG(Z,CIRX(NUM),CIRY(NUM),200.,0.,360.)
C TURN OFF ALL CIRCLES.
C CALL DISPLG(Z,2,8,ID(11))
C                               WAIT FOR AN ATTENTION TO OCCUR.
C 200 CALL WAITSG(Z,1,ICHECK,IVAL,X,Y)
```



```
C          IF LIGHT PEN ATTENTION, GO TO 210.
IF (ICHECK.EQ.1) GO TO 210
C          IF PROGRAMMED FUNCTION KEY ATTENTION, GO TO 235.
IF (ICHECK.EQ.3) GO TO 235
GO TO 200
C          IF LIGHT PEN HIT CENTER X, GO TO 221
210 IF (IVAL-ID(1)) 201,221,201
C          IF LIGHT PEN HIT OUTER X, GO TO 222
201 IF(IVAL-ID(9)) 222,222,202
C          IF LIGHT PEN HIT CIRCLE, GO TO 230
C          IF LIGHT PEN HIT LABEL, GO TO 225
202 IF (IVAL-ID(10)) 225,225,230
C          HERE IF CENTER X HIT.  TURN ON LABELS.
221 CALL DISPLG(2,3,1,ID(10))
GO TO 200
C          HERE IF OUTER X HIT.  TURN ON CIRCLE AROUND IT.
222 IVAL = IVAL + 9
CALL DISPLG(2,3,1,IVAL)
GO TO 200
C          HERE IF LABEL HIT, TURN OFF ALL LABELS.
225 CALL DISPLG(2,2,1,ID(10))
GO TO 200
C          HERE IF CIRCLE HIT.  TURN OFF CIRCLE.
230 CALL DISPLG(2,2,1,IVAL)
GO TO 200
C          HERE IF FUNCTION KEY HIT.  IF KEY 1 HIT, GO TO 240.
C          IF KEY 2, GO TO 250.  IF KEY 3, GO TO 260
235 IF (IVAL.GT.3) GO TO 200
GO TO (240,250,260),IVAL
C          HERE IF KEY 1 HIT.  TERMINATE.
240 CALL EXITG(2)
CALL EXIT
C          HERE IF KEY 2 HIT.  TURN OFF ALL CIRCLES.
250 CALL DISPLG(2,2,8,ID(11))
GO TO 200
C          HERE IF KEY 3 HIT.  TURN ON ALL CIRCLES.
260 CALL DISPLG(2,3,8,ID(11))
GO TO 200
END
```

/*

SAMPLE PROGRAM 3
PAL 49

/TEST JOB (8273,C),88750
/ EXC XFORTCLG
/FORT.SYSIN DD *

SUBROUTINE MAIN

THE FOLLOWING SAMPLE PROGRAM IS INTENDED TO INTRODUCE THE USER TO SOME OF THE CONCEPTS OF THE GRAPHICS SYSTEM. THE SAMPLE PROGRAM IS WRITTEN IN FORTRAN BUT IT COULD AS EASILY HAVE BEEN WRITTEN IN PL/I OR ASSEMBLY LANGUAGE. SEE THE INDIVIDUAL SUBROUTINE WRITTEUPS FOR A DESCRIPTION OF THE ARGUMENTS IN THE CALLING SEQUENCES.

THE DEMONSTRATION WILL BE SOMEWHAT SOPHISTICATED AS FAR AS GRAPHICS ARE CONCERNED, BUT SIMPLE IN THE CONCEPTS INVOLVED. THE USER WILL FIRST BE ASKED TO TRACE IN A MAP. THEN HE WILL BE REQUIRED TO PLACE CITIES ON THE MAP AND GIVE THEIR LONGITUDE AND LATITUDE. FINALLY HE WILL TRACE IN VARIOUS ROUTES HE WANTS TO TRAVEL AND THE DISTANCES AND TRAVEL TIMES WILL BE DISPLAYED.

```

*****INITIALIZATION*****
DIMENSION THE MODE SET ARRAY TO 200.
DIMENSION Z(200)
THEY CALL ON MODESG TO INITIALIZE THE SYSTEM AND START UP THE TABLET.
CALL MODESG(Z,I)
I AM NOW READY TO LET HIM TRACE IN THE MAP.
I MUST FIRST TELL HIM WHAT TO DO.
A CALL TO GETIDG IS MADE TO GET THE ID FOR THE TEMPORARY TEXT.
100 CALL GETIDG(Z,IDTGT)
THEN I CALL ON LEGNDG TO OUTPUT THE TEXT.
CALL LEGNDG(Z,1000.,4095.,16.,TRACE IN THE MAP')
NEXT, DISPLAY A LINE FOR HIM TO TOUCH TO TELL US WHEN HE IS DONE.
GET A NEW ID FOR IT.
CALL GETIDG(Z,IDBT)
CALL ON LEGNDG TO DISPLAY THE LINE.
CALL LEGNDG(Z,0.,50.,20.,TOUCH HERE WHEN DONE')

*****DRAW THE MAP*****
GET AN ID FOR THE MAP.
CALL GETIDG(Z,IDMAP)
WAIT FOR THE PEN TO GO DOWN.
1000 CALL WAITSG(Z,0,I,IVAL,X,Y)
PEN IS DOWN. SEE IF HE TOUCHED THE EXIT LINE. IF HE DID, GO TO 2000.
IF (IVAL.EQ.IDBT) GO TO 2000
OTHERWISE GET THE NEXT X,Y POSITION OF THE PEN.
1010 CALL GSTATG(Z,0,IVAL,X,Y)
GO WAIT FOR THE NEXT PEN DOWN IF THE PEN COMES UP.
IF (IVAL.EQ.D) GO TO 1000
OTHERWISE, PLOT THE POINT.
CALL POINTG(Z,I,X,Y)
AND GO FOR THE NEXT POINT.
GO TO 1010
WAIT FOR PEN TO COME BACK UP.
2000 CALL WAITSG(Z,0,0,IVAL,X,Y)
ERASE OLD MESSAGE.
CALL DISPLG(Z,D,1,IDTGT)
*****DRAW IN CITIES*****

```

```

C AND THEN PUT OUT A NEW MESSAGE.
CALL GETIDG(Z,IDTGT)
CALL LEGNDG(Z,D,4095.,54.,EXCELLENT: NOW PUT THE PEN DOWN WHERE
YOU WANT A CITY')
C ICITY WILL COUNT THE NUMBER OF CITIES HE DEFINES. (ALLOW 10)
ICITY = 0
C STORE CITY NAME IN ARRAY CITY. STORE LONGITUDE AND LATITUDE IN ALAT
C AND ALONG. STORE POSITION ON SCOPE IN XC,YC.
DIMENSION CITY(ID),ALAT(ID),ALONG(ID),XC(10),YC(10)
DIMENSION ACIT(10),BLAT(10),BLONG(10),XD(10),YD(10)
DATA ACIT/'LAX SFO SEA DEN DAL STL MOT DET MIA NYC '/
DATA BLAT/34.0,37.79,47.66,39.68,32.80,38.67,48.27,42.38,25.75,
140.66/
DATA BLONG/118.25,122.43,122.30,104.95,96.80,90.25,101.32,83.08,
180.25,73.83/
DATA XD/475.,256.,548.,1572.,2267.,2705.,1974.,3181.,3729.,3875./
DATA YD/2133.,2609.,3559.,2536.,1951.,2462.,3376.,2974.,1366.,
12901./
C STORE SOME DEFAULT VALUES IN CASE HE DOESN'T WANT TO BOTHER TO WRITE
C ANYTHING IN.
DO 2005 I = 1,10
CITY(I) = ACIT(I)
ALAT(I) = BLAT(I)
ALONG(I) = BLONG(I)
XC(I) = XD(I)
YC(I) = YD(I)
2005 YC(I) = YD(I)
C PUT THE 10'S FOR THE DIRECTIONS IN AN ARRAY SO WE CAN TREAT THEM AS
C ONE DISPLAY LATER.
DIMENSION IDARY(10)
IDARY(1) = IDTGT
IDARY(2) = IDBT
C WAIT FOR A PEN DOWN FOLLOWED BY A PEN UP
2010 CALL WAITSG(Z,0,1,IVAL,X,Y)
CALL WAITSG(Z,0,0,1,X,Y)
C SEE IF HE TOUCHED THE EXIT LINE. IF SO, GO TO 3000 TO DO NEXT PART.
IF (IVAL.EQ.IDBT) GO TO 3000
IF (ICITY.GE.10) GO TO 3000
C COUNT A CITY.
ICITY = ICITY + 1
XC(ICITY) = X
YC(ICITY) = Y
C TURN OFF THE OLD INSTRUCTIONS.
CALL DISPLG(Z,2,2,IDARY)
C PUT OUT NEW MESSAGE TELLING HIM WHAT TO DO.
CALL GETIDG(Z,IDT)
CALL LEGNDG(Z,0.,4095.,99.,WRITE IN THE CITY NAME, LONGITUDE, AND
LATITUDE. SWEEP THE PEN ACROSS THE TABLET WHEN YOU ARE DONE.)
C GET CITY NAME, LONGITUDE AND LATITUDE.
CALL DATAG(Z,1000.,3000.,3.,CITY = ',-4,CITY(ICITY),LONG = ',
110.4,ALONG(ICITY),LAT = ',10.4,ALAT(ICITY))
CALL DISPLG(Z,0,1,IDT)
C TURN OFF INSTRUCTIONS BACK ON.
CALL DISPLG(Z,1,2,IDARY)
C AND GO FOR NEXT CITY.
GO TO 2010
C
C
C

```

```
ERASE ALL UNWANTED TEXT.  
3000 CALL DISPLG(Z,0,1,IDTXT)  
IF HE DIDN'T SPECIFY ANY CITIES, GIVE HIM THE DEFAULT ONES.  
IF (ICITY.EQ.0) ICITY = 10  
CONVERT LONG & LAT TO RADIANS.  
DO 3010 I = 1,ICITY  
ALONG(I) = (ALONG(I)-90.)*0.01745329  
3010 ALAT(I) = ALAT(I)*0.01745329  
SET UP EACH CITY AS A BUTTON. EACH BUTTON WILL HAVE 4 CHARACTERS SO  
WE MUST MAKE A MODE SET CALL.  
CALL SETSMG(Z,95,4.)  
WE WANT TO SPECIFY THE X,Y OF EACH BUTTON SO WE NEED ANOTHER MODE S.T.  
CALL SETSMG(Z,94,1.)  
NOW WE CAN GENERATE THE BUTTONS.  
CALL GETIDG(Z,IOCIT)  
CALL BUTNG(Z,XC,YC,ICITY,CITY,1)
```

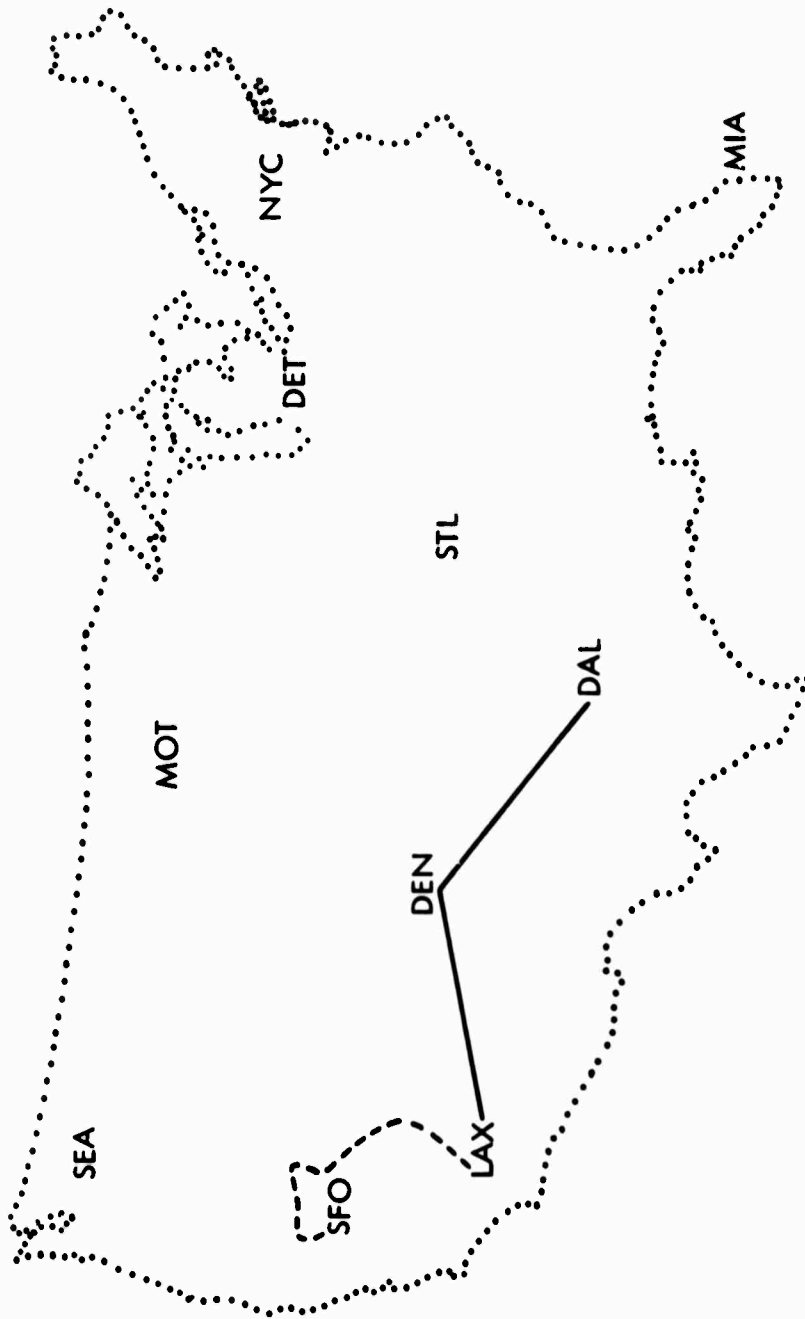
*****SET UP TO DRAW IN ROUTES*****

```
GIVE HIM THE OPTION OF STARTING OVER.  
CALL GETIDG(Z,IDOVR)  
CALL LEGNDG(Z,1200.,50.,24,'TOUCH HERE TO START OVER')  
PUT OUT THE EXPLANATION OF WHAT TO DO NEXT.  
CALL GETIDG(Z,IDTXT)  
IDARY(1) = IDTXT  
CALL LEGNDG(Z,0.,4095.,51,'FABULOUS: TRACE IN A ROUTE YOU WOULD LI  
IKE TO TRAVEL')  
DISPLAY DISTANCE AND TIME.  
CALL LEGNDG(Z,0.,300.,10,'DISTANCE =')  
CALL LEGNDG(Z,0.,400.,6,'TIME =')  
CALL GETIDG(Z,IDSP)  
CALL LEGNDG(Z,0.,200.,29,'WRITE SPEED, IN MPH, IN HERE:')  
DATA SPEED/1000./  
MAKE SPEED UNPROTECTED CHARACTERS SO WE CAN WRITE IN NEW VALUES.  
CALL SETSMG(Z,92,1.)  
CALL GETIOC(Z,IDN)  
CALL NUMBRG(Z,1726.,200.,10.3,SPEED)  
MAKE A SENSITIVE AREA AROUND THE SPEED.  
CALL AREASG(Z,1,1726.,150.,2236.,250.,11)  
PUT OUT NUMERIC VALUES FOR DISTANCE AND TIME.  
DATA TIME/0./,DIST/0./  
CALL GETIDG(Z,IDD)  
CALL NUMBRG(Z,650.,300.,10.3,DIST)  
CALL NUMBRG(Z,650.,400.,10.3,TIME)  
RESET TO PROTECTED DATA.  
CALL SETSMG(Z,92,0.)  
SET ID FOR ROUTE = 0.  
IDRT = 0  
GO TO 3020
```

*****TRACE IN THE ROUTE*****

```
ERASE ANY OLD INK.  
3015 CALL DISPLG(Z,0,1,IDINK)  
SET TIME AND DISTANCE BACK TO ZERO.  
3020 TIME = 0.  
DIST = 0.  
LCITY WILL TELL US WHICH CITY WE WERE JUST AT.  
LCITY = 0
```

```
C NOW WAIT FOR A PEN DOWN
  CALL WAITSG(Z,0,1,IVAL,X,Y)
C IF HE IS DONE, GO TO 5000.
  IF (IVAL.EQ.IDBT) GO TO 5000
C IF HE WANTS TO START OVER, GO TO 6000
  IF (IVAL.EQ.IDOVR) GO TO 6000
C ERASE ANY OLD ROUTING.
  CALL DISPLG(Z,0,1,IDRT)
C AND GET AN ID FOR A NEW ONE.
  CALL GETIDG(Z,IDRT)
C SEE IF HE TOUCHED A BUTTON.
  CALL PUSHG(Z,X,Y,ISEQ)
C IF HE WANTS TO CHANGE THE SPEED, GO TO 4000.
  IF (ISEQ.EQ.11) GO TO 4000
C OTHERWISE, POSITION THE BEAM TO THIS X,Y LOCATION.
3025 CALL GETIDG(Z,IDINK)
  CALL LINESG(Z,0,X,Y)
C WAS A CITY HIT? IF SO, GO TO 3050
3030 IF ((ISEQ.GE.1).AND.(ISEQ.LE.(LCITY))) GO TO 3050
C OTHERWISE, GET NEXT X,Y.
3040 CALL GSTATG(Z,0,IVAL,X,Y)
C IF PEN COMES UP, GO TO 3015 TO ERASE INK AND AWAIT PEN DOWN.
  IF (IVAL.EQ.0) GO TO 3015
C DRAW A LINE TO THIS X,Y
  CALL LINESG(Z,1,X,Y)
C SEE IF A CITY WAS HIT.
  CALL PUSHG(Z,X,Y,ISEQ)
  GO TO 3030
C HERE IF A CITY IS HIT. IS IT A NEW CITY? IF NOT, IGNORE IT.
3050 IF (ISEQ.EQ.LCITY) GO TO 3040
C IF IT IS THE FIRST CITY, GO ON.
  IF (LCITY.EQ.0) GO TO 3070
C
C *****COMPUTE TIME AND DISTANCE*****
C
C ERASE INK.
  CALL DISPLG(Z,0,1,IDINK)
C DRAW IN A STRAIGHT LINE SHOWING THE ROUTING.
  CALL SEGMTG(Z,1,XC(LCITY),YC(LCITY),XC(ISEQ),YC(ISEQ))
C COMPUTE NEW DISTANCE TRAVEL TIME.
  D = ABS(ALAT(ISEQ)-ALAT(LCITY))
  IF (D.LE.3.1415927) GO TO 3060
  D = 6.28319 - D
3060 D = ARCOS(COS(ALONG(LCITY))*COS(ALONG(ISEQ))*COS(D)+
  1*SIN(ALONG(LCITY))*SIN(ALONG(ISEQ)))*3441.594
  DIST = DIST + D
  TIME = TIME + D/SPEED
  DIMENSION CHARS(5)
C CONVERT NUMERIC DATA TO EBCDIC
  CALL FMTSG(Z,2,10,3,DIST,CHARS)
C AND REPLACE OLD VALUE ON SCOPE.
  CALL GPCHR(2,1,IDD,1,10,CHARS)
C DO THE SAME FOR TIME.
  CALL FMTSG(Z,2,10,3,TIME,CHARS)
  CALL GPCHR(7,1,IDD,11,10,CHARS)
  LCITY = ISEQ
C GO CHECK NEXT STOP ON THE ROUTE.
  GO TO 3025
3070 LCITY = ISEQ
```



Time = 1.495
Distance = 1494.820
Write speed, in MPH, in here: 1000.000

Touch here when done Touch here to start over

Fig. 11--Output from Sample Program 3

GO TO 3040

C
C*****WRITE IN NEW SPEED*****

C HERE TO CHANGE THE SPEED. TURN OFF UNWANTED TEXT.

4000 CALL DISPLG(Z,2,2, IDARY)

C TELL HIM HOW TO EXIT.

CALL GETIDG(Z, IDT)

CALL LEGNDG(Z,0.,4095.,49,'SWEEP THE PEN ACROSS THE TABLET WHEN YOU ARE DONE')

C RECOGNIZE THE CHARACTERS HE IS WRITING IN.

CALL CHARG(Z)

C RETRIEVE THE NEW SPEED.

CALL GPCHRG(Z,0, IDN,1,10, CHARS)

C CONVERT EBCDIC TO FLOATING POINT.

CALL CONVIG(Z,2,10, CHARS, SPEED)

C DELETE MESSAGE.

CALL DISPLG(Z,0,1, IDT)

C AND TURN OTHER MESSAGES BACK ON.

CALL DISPLG(Z,3,2, IDARY)

C GO TO 3020 TO CONTINUE

GO TO 3020

C
C*****EXIT*****

C HERE WHEN DONE.

5000 CALL EXITG(Z)

RETURN

C HERE TO START OVER.

6000 CALL DISPLG(Z,0,0,0)

GO TO 100

END

/*

Appendix B

IGS ERROR CODES

This table describes the meaning of each IGS error message. When an error is detected by a graphic subroutine, ERRZZ is called to print out an error message that reads as follows: BAD BAD BAD, ERROR NO. NO = VALUE(I Format) VALUE(F Format) VALUE(A Format). After the message is printed, the job continues in a normal manner.

Number	Subroutine	Value	Description
1	GETSMG	NO	Illegal mode set number in call.
2	LEGNDG	N	Illegal character count in call.
3	LINESG	NO	Illegal number in call.
4	METAZZ	-	2250 buffer is full.
5	NUMBRG	FMT	Illegal format in call.
6	OBJCTG	-	Max x or y LE min x or y in call.
7	METAZZ	-	No call to GETIDG.
8	POINTG	N	Illegal number in call.
9	SEGMTG	N	Illegal number in call.
10	DATAG	-	Too many arguments in call.
11	MLTPLG	NLINES	Illegal number in call.
12	GRIDG	-	Grid too small to draw.
13	LABELG	-	Illegal format in call.
14	TITLEG	-	Illegal arguments in call.
15	SETUPG	-	Illegal arguments in call.
16	SUBJEG	-	Max x or y EQ min x or y.
17	LABELG	-	Illegal arguments in call.
18	LABELG	-	Grid too small to label.
19	LABELG	-	Zero subject space.
20	GRIDG	-	Illegal arguments in call.
21	SETUPG	-	Subject (object) space too small to draw a grid.
22	SETUPG	-	Density LE 0.
23	SETSMG	N	Illegal mode set number in call.
24	SETUPG	-	Grid will not fit on page.
25	TEXTG	N	Illegal character count in call.
26	LABELG	-	Labels will not fit on page.
27	METAZZ	-	No initialization call to MODESG.
28	GRAPHG	N	Illegal argument in call.
29	SUBJEG	-	Minus value for log grid.
30	SETUPG	-	Too many cycles in log grid.
32	SCALZZ	X	Bad x-coordinate.
33	SCALZZ	Y	Bad y-coordinate.
35	TITLEG	-	Subject (object) space too small to title grid.

Appendix C

SUMMARY OF 2250 GRAPHIC FEATURES

Item	Value
Screen size	12" x 12"
Raster size	4095 x 4095
Character set	63 characters
	A-Z 0-9
	¢) : . \$, #
	< * % ' @
	() - +
	; > = /
	? " -
Basic character size	1.0 x normal
Number lines/page	52
Number characters/line	74
Character spacing (envelope width)	56 raster units
Line spacing (envelope height)	80 raster units
Large character size	1.5 x normal
Number lines/page	35
Number characters/line	49
Character spacing (envelope width)	84 raster units
Line spacing (envelope height)	120 raster units

Appendix D

CORE STORAGE ALLOCATION

All FORTRAN subroutines are compiled under Level-G FORTRAN.
Storage is given in bytes.

NAME	DEC	HEX	NOTES
ALARMG	158	9E	
AREASG	806	326	
BUTNG	950	3B6	
CHANG	240	F0	
CHARG	706	2C2	
CIRARG	1718	6B6	
CONVTG	1520	5F0	
CURSRG	620	26C	
DATAG	2986	8AA	
DEBUG	912	390	
DISPLG	1058	422	
EXITG	240	F0	
FMTSG	2536	9E8	
GETIDG	232	E8	
GETSMG	534	216	
GPCHRG	940	3AC	
GPXYG	1672	688	
GRAPHG	1364	554	
GRIDG	4054	FD6	
GSTATG	564	234	
GTIMEG	248	F8	
LABELG	4228	1084	
LEGNDG	516	204	
LINESG	686	2AE	
MLTPLG	886	376	
MODESG	432	180	
NUMBRG	1306	51A	
OBJCTG	1208	488	
POINTG	504	1F8	
PUSHG	584	248	
RECOG	2128	850	
RSETMG	1028	404	
SEGMTG	536	218	
SETFKG	412	19C	
SETUPG	4280	10B8	
SETSMG	1642	66A	
SNAPG	1440	5A0	
SUBJEG	1582	62E	
TEXTG	472	1D8	
TITLEG	2660	A64	
WAITSG	748	2EC	
WTIMEG	220	DC	
BUFFZZ	204	CC	
CBUTZZ	420	1A4	
CHARZZ	13700	3584	
CTABZZ	532	214	
C225ZZ	888	378	NOT RESIDENT-LINK MACRO USED FOR CALL.
ERRZZ	380	17C	
FRUTZZ	548	224	
GETCZZ	160	A0	

NOT RESIDENT-LINK MACRO USED FOR CALL.
NOT RESIDENT-LINK MACRO USED FOR CALL.

GNXYZZ	932	3A4
IFMZZ	372	174
METAZZ	2316	406
PACKZZ	0	0
PUTCZZ	158	9E
PZZ	468	104
REPZZ	646	286
SCALZZ	1186	4A2
TABLZZ	866	362
TRAPZZ	700	236
UNSCZZ	744	2E8
XMODZ	290	122
XNORMZ	308	134
YMODZ	290	122
YNORMZ	308	134

SHOULD BE IN MAIN LINK OF AN OVERLAY.
INCLUDED IN SNAPG.

NOT RESIDENT-LINK MACRO USED FOR CALL.
SHOULD BE IN MAIN LINK OF AN OVERLAY.
MUST BE IN MAIN LINK OF AN OVERLAY.
MUST BE IN MAIN LINK OF AN OVERLAY.

Appendix E

SUBROUTINE CROSS-REFERENCE LIST

NAME	USES	USED BY
ALARMG		
AREASG	CBUTZZ,LINESG	
BUTNG	CBUTZZ,LEGNDG,GNXYZZ	
CHANG		
CHARG	CHARZZ	DATAG
CIRARG	LINESG,POINTG	RECOG
CONVTG	GETCZZ	DATAG
CURSRG		DATAG
DATAG	ERRZZ,GETIDG,CHARG, LEGNDG,GPCHRG,NUMBRG, CONVTG,GNXYZZ,DISPLG, CURSRG	
DEBUG	GETIDG,NUMBRG,DISPLG,WAITSG	
DISPLG		DEBUG,DATAG
EXITG	C225ZZ,CTABZZ	
FMTSG	IFMZZ,GETCZZ,PUTCZZ	NUMBRG
GETIDG		DEBUG,DATAG
GETSMG	ERRZZ	
GPCHRG		DATAG
PXYG	SCALZZ,UNSCZZ	
GRAPHG	SUBJEG,SETUPG,GRIDG,LABELG, TITLEG,POINTG,ERRZZ	
GRIDG	SETSMG,MLTPLG,SEGMTG, ERRZZ	GRAPHG
ISTATG	UNSCZZ	
TIMEG		
LABELG	SETSMG,MLTPLG,SEGMTG, NUMBRG,ERRZZ	GRAPHG
LEGNDG	METAZZ,ERRZZ,TEXTG	BUTNG,NUMBRG,LABLEG,TITLEG, DATAG
LINESG	METAZZ,ERRZZ	AREASG,CIRARG,RECOG
MLTPLG	SEGMTG,ERRZZ	GRIDG,LABELG
MODESG	C225ZZ,CTABZZ,RSETMG	
NUMBRG	LEGNDG,FMTSG,ERRZZ	DEBUG,LABELG,DATAG
OBJCTG	ERRZZ	SETUPG
POINTG	METAZZ,ERRZZ	CIRARG,GRAPHG
PUSHG	WAITSG,FBUTZZ	
RECOG	CHARZZ,LINESG,CIRARG, SEGMTG,UNSCZZ	
RSETMG		MODESG
SEGMTG	METAZZ,ERRZZ	RECOG,GRIDG,PARALG
SETFKG		
SETUPG	SUBJEG,OBJCTG,ERRZZ	GRAPHG
SETSMG	ERRZZ	GRIDG,TITLEG
SNAPG	PACKZZ	TRAPZZ
SUBJEG	XMODZ,YMODZ,ERRZZ	SETUPG,GRAPHG,LABELG
EXITG	LEGNDG,ERRZZ	
TITLEG	SETSMG,LEGNDG,ERRZZ	GRAPHG
WAITSG	UNSCZZ	DEBUG,PUSHG
TIMEG		
UFFZZ		
BUTZZ	SCALZZ	AREASG,NUMBRG

CHARZZ	CHARACTER RECOGNIZER	CHARG, RECOG
CTABZZ	TABLZZ	EXITG, MODESG
C225ZZ	TABLZZ, TRAPZZ	EXITG, MODESG
ERRZZ		ALMOST ALL
FBUTZZ	SCALZZ	PUSHG
GETCZZ		CONVTG, FMTSG
GNXYZZ		BUTNG, DATAG
IFMZZ		FMTSG
METAZZ	UNSCZZ, ERRZZ, REPKZZ,	LEGNDG, LINESG, POINTG,
	SCALZZ	SEGMTG
PACKZZ	ERRZZ	SNAPG
PUTCZZ		CONVTG, FMTSG
PZZ		
REPKZZ		METAZZ
SCALZZ	XMODZ, YMODZ, ERRZZ	GPXYG, CBUTZZ, FBUTZZ,
		METAZZ
TABLZZ		C225ZZ, TRAPZZ
TRAPZZ	SNAPG, TABLZZ	C225ZZ, CTABZZ
UNSCZZ		GPXYG, GSTATG, METAZZ, RECOG,
		WAITSG
XMODZ		SCALZZ, SUBJEG
XNORMZ		
YMODZ		SCALZZ, SUBJEG
YNORMZ		

Appendix F
JOB CONTROL LANGUAGE

The job control statements need several special additions to run graphic jobs. First, the user must concatenate the graphic library with his normal library. If he is executing PL/I or assembly language, he must also concatenate the FORTRAN library. Two INCLUDE cards are required to ensure the loading of resident IGS routines. In addition, the user must provide four cards at the end of his decks to define the graphic data sets. The job control statements needed are as follows:

```
//jobname JOB (appropriate operands)
//[stepname] EXEC (whatever is being executed)
//LKED.SYSLIB DD DSNAME=SYS1.GRAPLIB,DISP=SHR
//           DD DSNAME=SYS1.FORTLIB,DISP=SHR
//           DD DSNAME=SYS1.LINKLIB,DISP=SHR
//LKED.SYSIN DD *
  INCLUDE SYSLIB(TRAPZZ)
  INCLUDE SYSLIB(TABLZZ)

  [object modules]

/*
//GO.GRAPHIC DD UNIT=2250
//GO.TABLET DD UNIT=TABLET
//GO.IGSLINK DD DSNAME=SYS1.GRAPLIB,DISP=SHR
//GO.SC4060ZZ DD DSNAME=SCOOPS,DISP=(MOD,PASS)
//GO.SYSIN DD *
  [any data cards]
/*
```

The two following special procedures have been added to the procedures library for the FORTRAN programmer.

- 1) To compile, link edit, and execute:

```
//jobname JOB (appropriate operands)
//[stepname] EXEC XFORTCLG
//FORT.SYSIN DD *
    [FORTRAN source decks]
/*
//LKED.SYSIN DD *
    [any object decks] } (Include only if there
/*
//GO.SYSIN DD *
    [any data cards] } (Include only if data
/*
                        } are read in from the
                        } card reader.)
```

2) To link edit and execute previously compiled decks:

```
//jobname JOB (appropriate operands)
//[stepname] EXEC XFORTLG
//LKED.SYSIN DD *
    [object decks]
/*
//GO.SYSIN DD *
    [any data cards] } (Include only if data
/*
                        } are read in from the
                        } card reader.)
```

Two additional procedures assist the PL/I programmer:

3) To compile, link edit, and execute:

```
//jobname JOB (appropriate operands)
//[stepname] EXEC GPLIFCLG
//PL1L.SYSIN DD *
    [PL/I source statements]
/*
//LKED.SYSIN DD *
    [any object decks] } (Include only if there
/*
//GO.SYSIN DD *
    [any data cards] } (Include only if data
/*
                        } are read in from the
                        } card reader.)
```

4) To link edit and execute previously compiled decks:

```
//jobname JOB (appropriate operands)
//[stepname] EXEC GPL1LG
//LKED.SYSIN DD *
    [object decks]
/*
//GO.SYSIN DD *
    [any data cards] } (Include only if data
/*
                        } are read in from the
                        } card reader.)
```

Appendix G
PL/I COMPATIBILITY

IGS is written primarily to be called from FORTRAN; however, it may be called from PL/I with a few changes in procedure necessary because of differences between the languages.

IGS must be informed with the following mode set call that it will be called by PL/I:

```
A = 1.0,  
I = 80;  
CALL SETSMG(Z(1),I,A);
```

In FORTRAN, the array name and the first element of the array are equivalent; in PL/I, they differ. Thus, a PL/I call to IGS must specifically refer to the first element of an array rather than the array name. Literals cannot be used in PL/I calls; only variables. Thus a call that would be

```
CALL SETSMG(Z,80,1.)
```

in a FORTRAN program must be

```
A = 1.0;  
I = 80;  
CALL SETSMG(Z(1),I,A);
```

in a PL/I program. PL/I also differs from FORTRAN in that double-subscripted arrays are stored in row order rather than column order. Double-subscripted PL/I arrays must be transposed for calls to IGS.

The PL/I error routine that handles abnormal terminations will itself abnormally terminate if the original termination does not come from a PL/I subroutine. Since IGS is not written in PL/I, any abnormal terminations within IGS will be lost by the subsequent termination of the

PL/I error routine. The indicative dump will show only the PL/I error routine termination with no indication that the termination originated with IGS. It is hoped that there will be few abnormal terminations within IGS.

Appendix H

IGS MODE SET ARRAY

This section describes the contents of the mode set array. SETSMG does most mode setting, but RSETMG, SUBJEG, and OBJCTG can also do it.

NO	FUNCTION	DESCRIPTION	DEFAULT VALUE	SET BY
1	INITIALIZATION	TELLS WHETHER ARRAY IS INITIALIZED.	-	SYSTEM
2	MIN-X	SUBJECT SPACE.	0.	SUBJEG
3	MIN-Y	SUBJECT SPACE.	0.	SUBJEG
4	MAX-X	SUBJECT SPACE.	4095.	SUBJEG
5	MAX-Y	SUBJECT SPACE.	4095.	SUBJEG
6	MIN-X	NORMALIZED OBJECT SPACE	0.	OBJCTG
7	MIN-Y	NORMALIZED OBJECT SPACE	0.	OBJCTG
8	MAX-X	NORMALIZED OBJECT SPACE	1.	OBJCTG
9	MAX-Y	NORMALIZED OBJECT SPACE.	1.	OBJCTG
10	X-OFFSET	SCALING FACTORS COMPUTED BY SYSTEM.	0.	SYSTEM
11	Y-OFFSET	SCALING FACTORS COMPUTED BY SYSTEM.	0.	SYSTEM
12	X-SCALE	SCALING FACTORS COMPUTED BY SYSTEM.	1.	SYSTEM
13	Y-SCALE	SCALING FACTORS COMPUTED BY SYSTEM.	1.	SYSTEM
14	SCALE FLAG	TYPE OF SCALING TO DO. 0.-REAL SUBJECT SPACE UNITS. 1.-REAL ABSOLUTE RASTER UNITS. 2.-INTEGER ABSOLUTE RASTERS. 3.-REAL NORMALIZED OBJECT SPACE.	0.	SYSTEM
15	RESERVED			
16	RESERVED			
17	LAST X	CURRENT POINT POSITION IN SUBJECT SPACE.	-	SYSTEM
18	LAST Y	CURRENT POINT POSITION IN SUBJECT SPACE.	-	SYSTEM
19	X-NORM. FACTOR	SCOPE SIZE IN RASTERS/MAX NORM.	4095.	SETSMG
20	Y-NORM. FACTOR	SCOPE SIZE IN RASTERS/MAX NORM.	4095.	SETSMG
21	MAX-X	SCOPE SIZE-IN ABSOLUTE RASTER UNITS	4095.	SETSMG
22	MAX-Y	SCOPE SIZE-IN ABSOLUTE RASTER UNITS	4095.	SYSTEM
23	X-LINEAR FLAG	FLAG FOR LINEAR SCALING IN X-AXIS. 0.-LINEAR 1.-NONLINEAR	0.	SETSMG
24	Y-LINEAR FLAG	FLAG FOR LINEAR SCALING IN Y-AXIS. 0.-LINEAR 1.-NONLINEAR	0.	SETSMG
25	LOG(MIN-X)	LOG OF SUBJECT SPACE.	-	SYSTEM
26	LOG(MIN-Y)	LOG OF SUBJECT SPACE.	-	SYSTEM
27	LOG(MAX-X)	LOG OF SUBJECT SPACE.	-	SYSTEM
28	LOG(MAX-Y)	LOG OF SUBJECT SPACE.	-	SYSTEM
29	VACANT			
30	RESERVED			
31	RESERVED			
32	VACANT			
33	VACANT			
39	VACANT			
40	NORMAL WIDTH	NORMAL CHARACTER WIDTH IN NORMALIZED OBJECT SPACE.	.01367	SYSTEM

41	NORMAL HEIGHT	NORMAL CHARACTER HEIGHT IN NORMALIZED OBJECT SPACE.	.01953	SYSTEM
42	CURRENT WIDTH	CHARACTER WIDTH IN SUBJECT SPACE.	56.	SETSMG
43	CURRENT HEIGHT	CHARACTER HEIGHT IN SUBJECT SPACE.	80.	SETSMG
44	RESERVED			
45	CHARACTER SIZE	NO. TIMES NORMAL SIZE FOR TYPING. 1.0 56X80 RASTER UNITS. 1.5 84X120 RASTER UNITS.	1.	SETSMG
46	RESERVED			
47	RESERVED			
48	VACANT			
49	LINE SPACING	NO. TIMES CURRENT CHARACTER HEIGHT TO SPACE LINE UPON EJECTION. + VALUE, SPACE LINE UP. - VALUE, SPACE LINE DOWN.	-1.	SETSMG.
50	LINE ORIENTATION	ANGLE IN DEGREES AT WHICH A LINE OF CHARACTERS IS TO BE DISPLAYED. 0. IS HORIZONTAL, TO THE RIGHT. COUNTER CLOCKWISE ANGLES ARE POSITIVE. 0. AND +-90. ARE ALLOWED FOR THE 2250.	0.	SETSMG
51	RESERVED			
52	RESERVED			
53	RESERVED			
54	RESERVED			
55	RESERVED			
56	VACANT			
	• VACANT			
59	VACANT			
60	LEFT MARGIN	MARGIN IN SUBJECT SPACE.	0.	SETSMG
61	RIGHT MARGIN	MARGIN IN SUBJECT SPACE.	4095.	SETSMG
62	BOTTOM MARGIN	MARGIN IN SUBJECT SPACE.	0.	SETSMG
63	TOP MARGIN	MARGIN IN SUBJECT SPACE.	4095.	SETSMG
64	CHARACTERS/LINE	NO. CHARACTERS PER LINE.	74.	SYSTEM
65	LINES/PAGE	NO. LINES PER PAGE.	52.	SYSTEM
66	RESERVED			
	•			
76	RESERVED			
77	FORMAT WIDTH	FORMAT WIDTH-USED BY NUMBRG.	-	SETSMG
78	FORMAT DECIMALS	FORMAT DECIMAL-USED BY NUMBRG.	-	SETSMG
79	FORMAT	FORMAT-USED BY NUMBRG 1. - I FORMAT 2. - F FORMAT 3. - E FORMAT 4. - A FORMAT	-	SETSMG
80	PL/I FLAG	PL/I CALLS. 0.-NO 1.-YES	0.	SETSMG
81	CHARACTER ADDRESS	N TH CHARACTER OF A STRING TO BEGIN ADDRESSING WITH.	1.	SETSMG
82	TYPE OF ARC	METHOD TO USE IN DRAWING AN ARC. 0.-LINES 1.-POINTS	0.	SETSMG
83	ROUNDNESS OF ARC	DEGREES BETWEEN TWO ADJACENT POINTS OF AN ARC.	5.	SETSMG
84	PLOT CHARACTER	CHARACTER TO USE FOR PLOTTING. (LEFT-JUSTIFIED)	POINT	SETSMG

86	VACANT		
87	BLOCK	ADDRESS OF RESIDENT BLOCK.	- SYSTEM
88	PEN UP DELAY	PEN UP DELAY IN SECONDS.	0.5 SETSMG
89	DX-DATAG	DX FOR SUBROUTINE DATAG.	0. SETSMG
90	DY-DATAG	DY FOR SUBROUTINE DATAG.	-80. SETSMG
91	ARRAY-DATAG	ARRAY FLAG FOR SUBROUTINE DATAG. 0.-NO ARRAY 1.-X,Y ARE ARRAYS	0. SETSMG
92	PROTECTION FLAG	PROTECTION FLAG FOR GRAPHIC OUTPUT. 0.-PROTECTED 1.-NOT PROTECTED	0. SETSMG
93	SIZE-DATAG	CHARACTERS PER LABEL FOR DATAG.	8. SETSMG
94	ARRAY-BUTTON	ARRAY FLAG FOR SUBROUTINE BUTTON. 0.-NO ARRAY 1.-X,Y ARE ARRAYS	0. SETSMG
95	BUTTON SIZE	NO. CHARACTERS/BUTTON.	8. SETSMG
96	DX-BUTTON	DX FOR SUBROUTINE BUTTON.	0. SETSMG
97	DY-BUTTON	DY FOR SUBROUTINE BUTTON.	-80. SETSMG
98	BOX-AREASG	FLAG FOR DRAWING A BOX FOR AREASG. 0.-NO BOX 1.-DRAW BOX	0. SETSMG
99	RESERVED		
100	AXES FLAG	EMPHASIS OF MAJOR AXES. 0.-EMPHASIZE BOTH 1.-EMPHASIZE X=0 ONLY 2.-EMPHASIZE Y=0 ONLY 3.-NO EMPHASIS	0. SETSMG
101	VACANT		
102	X TICK MARKS	LENGTH OF X-AXIS TICK MARK IN NORMALIZED OBJECT SPACE.	0. SETSMG
103	Y TICK MARKS	LENGTH OF Y-AXIS TICK MARK IN NORMALIZED OBJECT SPACE.	0. SETSMG
104	X-AXIS LABEL	NO. TIMES CHARACTER HEIGHT TO POSITION X-AXIS LABEL. + LABEL ABOVE GRID - LABEL BELOW GRID 0. LABEL AT Y=0	-1. SETSMG
105	Y-AXIS LABEL	NO. TIMES CHARACTER WIDTH TO POSITION Y-AXIS LABEL. + LABEL RIGHT SIDE OF GRID - LABEL LEFT SIDE OF GRID 0. LABEL AT X=0	-1. SETSMG
106	LEFT MARGIN	RELATIVE GRID MARGIN IN NORMALIZED OBJECT SPACE.	0.15 SETSMG
107	RIGHT MARGIN	RELATIVE GRID MARGIN IN NORMALIZED OBJECT SPACE.	0.15 SETSMG
108	BOTTOM MARGIN	RELATIVE GRID MARGIN IN NORMALIZED OBJECT SPACE.	0.15 SETSMG
109	TOP MARGIN	RELATIVE GRID MARGIN IN NORMALIZED OBJECT SPACE.	0.15 SETSMG
110	SQUARE GRID	FORCE VALUES FOR A SQUARE GRID TO BE CALCULATED. 0.-NOT FORCED TO BE SQUARE. 1.-MUST BE SQUARE.	0. SETSMG
111	X-AXIS DENSITY	MIN DISTANCE BETWEEN X-AXIS GRID LINES IN NORMALIZED OBJECT SPACE.	.01009SETSMG
112	Y-AXIS DENSITY	MIN DISTANCE BETWEEN Y-AXIS GRID LINES IN NORMALIZED OBJECT	.01660SETSMG

113 MIN X-TITLE	SPACE. NORMALIZED OBJECT SPACE UNAVAILABLE FOR TITLES.	-	LABELG
114 MIN Y-TITLE	NORMALIZED OBJECT SPACE UNAVAILABLE FOR TITLES.	-	LABELG
115 MAX X-TITLE	NORMALIZED OBJECT SPACE UNAVAILABLE FOR TITLES.	-	LABELG
116 MAX Y-TITLE	NORMALIZED OBJECT SPACE UNAVAILABLE FOR TITLES.	-	LABELG
117 VACANT			
. VACANT			
130 VACANT			
131 RESERVED			
. RESERVED			
150 RESERVED			
151 VACANT			
. VACANT			
200 VACANT			

Appendix I

HARDCOPY REQUESTS

A user may request hardcopy through subroutine calls within his program, or through special function keys at the 2250 console. Subroutine calls are convenient for large or specific requests. The subroutine calls for obtaining hardcopy are described in the write-up of SNAPG (p. 104). The function keys are more expedient for small requests, or if the user either does not want to reprogram or does not know in advance that he wants hardcopy. Hardcopy through the function keys is always available to any 2250 IGS program-- at no cost in core storage if unused. The two methods cannot be mixed. For either the subroutine-call or function-key method, open the hardcopy file, optionally reset for 8 1/2 x 11 hardcopy (11 x 14 is the default hardcopy size), initiate hardcopy of the 2250 displays, and finally close the file.

The function-key method works as follows:

- 1) Place a function-key overlay on the keys with notch 7 intact. This overlay preempts keys 30, 31, and 32 for hardcopy. The audio alarm will sound each time one of these keys is hit to inform the user that his request is recognized. The overlay may be removed at any time to allow normal use of keys 30, 31, and 32.
- 2) Strike function key 32. The first strike loads the hardcopy subroutine, initializes the hardcopy file, lights up keys 30, 31, and 32 as a reminder that the hardcopy file is active, and makes a copy of the current 2250 display. Each successive strike makes another copy of the 2250 display.

- 3) Key 30 controls the hardcopy size. Each strike switches the size. Since the default hardcopy size is 11×14 , the first strike changes the size to $8 \frac{1}{2} \times 11$. The second will switch the mode back to 11×14 , etc. The user may hit key 30 before key 32 if he wants all $8 \frac{1}{2} \times 11$ output. The light in key 30 will be on for 11×14 hardcopy and off for $8 \frac{1}{2} \times 11$.
- 4) Strike key 31 to close the hardcopy file when finished. This deletes the hardcopy subroutine, prints an online message on the 360/40 log indicating that hardcopy has been requested, and restores the function key lights to their original status.
- 5) Repeat steps 1 through 4 as often as desired during a job.

Appendix J
SYSTEM SUBROUTINES

IGS uses the following subroutines. The user is normally not concerned with them; they are included only for completeness of documentation.

- 1) BUFFZZ Changes the size of the internal buffer used to write to the 2250.
- 2) CBUTZZ Creates a control entry and files it in a list.
- 3) CHARZZ Recognizes handwritten characters.
- 4) CTABZZ Turns the RAND Tablet on and off.
- 5) C225ZZ Turns the IBM 2250 on and off.
- 6) ERRZZ General error subroutine.
- 7) FBUTZZ Detects a button touch.
- 8) GETCZZ/PUTCZZ Pack or unpack bytes.
- 9) GNXYZZ Computes the next x,y location of a button.
- 10) IFMZZ Converts an integer number to a character string.
- 11) METAZZ Writes graphic commands.
- 12) PACKZZ Packs and outputs 2250 hardcopy.
- 13) PZZ Prints the mode set array.
- 14) REPKZZ Reracks the 2250 buffer.
- 15) SCALZZ Converts user coordinates to rasters.
- 16) TABLZZ Resident data section.
- 17) TRAPZZ Processes 2250 input traps.
- 18) UNSCZZ Converts rasters to user coordinates.

BUFFZZ: Changes the Internal Buffer Size

PURPOSE

Alters the size of the internal IGS buffer used in writing to the 2250. A large buffer will write displays faster, but requires more core storage. The default buffer size is 600 bytes.

USAGE

Standard Calls

CALL BUFFZZ(Z,NO)

- 1) Z is the mode set array.
- 2) NO is the number of bytes at which to set the buffer size. The default value of NO is 600 bytes.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. BUFFZZ can be called only after the 2250 is initialized.

CBUTZZ: Creates, Then Files, a Control Entry

PURPOSE

Called by BUTTNG and AREASG to create and file control areas into a list. PUSHG searches this list to determine if a control area was touched.

USAGE

Standard Calls

CALL CBUTZZ(Z,XL,YL,XR,YR,ISEQ)

- 1) Z is the mode set array.
- 2) XL,YL is the lower left point of the control area.
- 3) XR,YR is the upper right point of the control area.
- 4) ISEQ is the sequence number to assign to the control area.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

CHARZZ: Recognizes Handwritten Characters

PURPOSE

Recognizes characters handwritten on the RAND Tablet. The subroutine returns after the pen has been lifted from the Tablet. The user may force a return by hitting the END key or a function key, or by using the light pen.

USAGE

Standard Calls

CALL CHARZZ(Z,IDATA,I,CHAR)

- 1) Z is the mode set array.
- 2) IDATA will contain the following data on return:
 - +IDATA(1) = x location of pen down (in integer rasters).
 - +IDATA(2) = y location of pen down (in integer rasters).
 - +IDATA(3) = x location of pen up (in integer rasters).
 - +IDATA(4) = y location of pen up (in integer rasters)
 - IDATA(5) = x location of centroid (in integer rasters).
 - IDATA(6) = y location of centroid (in integer rasters).
 - IDATA(7) = DX of character (in integer rasters).
DX = 4096 (integer) if the return was caused by the END key, function keys, or light pen.

†Applies only to the last stroke.

IDATA(8) = DY of character (in integer rasters).

IDATA(9) = Character (left-justified) or type of geometric figure (integer). A question mark (?) is returned if the character is unrecognizable.

Characters:

A through Z

0 through 9

. , + - * / = \$ () ' > < ε | → blank.

Geometric figure:

0-line

1-rectangle

2-circle

3-triangle

4-ellipse

5-diamond

6-trapezoid

IDATA(10) = Number of corners.

IDATA(11) = Type of return.

= 0, no character pending.

> 0, a character is pending.

3) I specifies the ink size to use.

If I = 192, ink size = 2 rasters.

= 196, ink size = 4 rasters.

= 198, ink size = 6 rasters.

= 200, ink size = 8 rasters.

4) CHAR contains the character recognized upon return.

Modifications

To change the pen-up delay time (time required for IGS to decide whether two pen strokes constitute a single character),

CALL SETSMG(Z,88,TIME)

TIME specifies the time in seconds to allow the user to delay in writing multi-stroke characters.
TIME = 0.5 is the default value.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. References 3 and 4 completely describe the character recognition routines. CHARZZ depends on the RAND Tablet.

CTABZZ: Turns the RAND Tablet On and Off

PURPOSE

Turns the RAND Tablet on or off.

USAGE

Standard Calls

CALL CTABZZ(Z,I)

- 1) Z is the mode set array.
- 2) I specifies the action to be taken.

If I = 0, start the Tablet up initially.

- = 1, close the Tablet down permanently.
- = 2, start the Tablet.
- = 3, stop the Tablet.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. CTABZZ depends on the RAND Tablet.

C225ZZ: Turns the IBM 2250 On and Off

PURPOSE

Turns the 2250 on or off.

USAGE

Standard Calls

CALL C225ZZ(Z,I)

- 1) Z is the mode set array.
- 2) I specifies the action to be taken.

If I = 0, turn the 2250 on.

▪ 1, turn the 2250 off.

▪ 2, release all core used for accounting.

▪ 3, initialize core for accounting.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

ERRZZ: General Error Subroutine

PURPOSE

IGS subroutines call ERRZZ whenever they detect a programmer error. ERRZZ prints out a message indicating the type of error. Appendix G (p. 125) lists the possible error messages.

USAGE

Standard Calls

CALL ERRZZ(Z,NO,VALUE)

- 1) Z is the mode set array.
- 2) NO is the error identification number.
- 3) VALUE is any variable associated with the error.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

FBUTZZ: Detects a Button Touch

PURPOSE

PUSHG calls FBUTZZ to determine whether a button contains an x,y coordinate. If so, FBUTZZ displays an asterisk beside the button, and returns the button's sequence number.

USAGE

Standard Calls

CALL FBUTZZ(Z,X,Y,ISEQ)

- 1) Z is the mode set array.
- 2) X,Y are x,y coordinates.
- 3) ISEQ contains the sequence number of the touched button, or zero if no button was touched.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

GETCZZ and PUTCZZ: Pack or Unpack Bytes

PURPOSE

Pack and unpack bytes.

USAGE

Standard Calls

- 1) To unpack a byte,

CALL GETCZZ(Z,BYTE,NTH,STRING)

- a) Z is the mode set array.
- b) BYTE contains the byte on return. The byte will be right-justified with leading positions set to zero.
- c) NTH specifies the byte to unpack from STRING. NTH = 1 indicates the first byte in STRING.
- d) STRING is a variable or array containing the bytes.

- 2) To pack a byte,

CALL PUTCZZ(Z,BYTE,NTH,STRING)

- a) Z is the mode set array.
- b) BYTE contains the right-justified byte to pack. The left portion of BYTE may contain anything.
- c) NTH specifies the position in STRING at which to store the byte. NTH = 1 indicates that the byte is to be stored in the first position.
- d) STRING is a variable or array containing the bytes.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

GNXYZZ: Computes the Next X,Y Location of a Button

PURPOSE

Computes where on the 2250 screen to position the next of a series of buttons.

USAGE

Standard Calls

CALL GNXYZZ(Z,X,Y,CHARP,CHAR,DX,DY)

- 1) Z is the mode set array.
- 2) X,Y is the x,y location of the previous button on entry. On return, it is set to the x,y location of the next button.
- 3) CHARP is the number of characters in the previous button.
- 4) CHAR is the number of characters in the next button.
- 5) DX,DY is the relative distance between the buttons.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

IFMZZ: Converts an Integer Number into a Character String

PURPOSE

Converts an integer number into a character string.

USAGE

Standard Calls

CALL IFMZZ(Z,NUMB,OUT,IW,ISTART,NFLG,IERR)

- 1) Z is the mode set array.
- 2) NUMB is the integer number to convert or the fill character if NFLG = -1.
- 3) OUT contains the character string on return. OUT must be large enough to contain IW characters.
- 4) IW specifies the total number of character positions available for output.
- 5) ISTART designates the leftmost character position in the output string (ISTART \geq 1).
- 6) NFLG stipulates the specific function to perform.
 - If NFLG = -1, fill the output string with the rightmost character in NUMB.
 - = 0, convert the number and fill positions to the left of the high-order digit with blanks.
 - = 1, convert the number and fill high-order positions with zeros.
- 7) IERR is an error return flag set by the routine.
 - If IERR = 0, the function was performed satisfactorily.
 - = 1, the field width (IW) was too small for the number to fit, so the output area will be filled with asterisks.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

METAZZ: Writes Graphic Commands

PURPOSE

Scales x,y coordinates, formats the data into 2250 buffer commands, and packs, buffers, and transmits the data to the 2250 buffer.

USAGE

Standard Calls

- 1) To plot a specific character,

CALL METAZZ(Z,1,N,X,Y,1,CHAR)

- a) Z is the mode set array.
- b) N is the number of points ($N \geq 1$).
- c) X,Y are arrays of N x,y coordinate pairs to plot.
- d) CHAR contains the left-justified plotting character to use for plotting. If CHAR = 0.0, normal-sized plotting points are used.

- 2) To type characters,

CALL METAZZ(Z,5,1,X,Y,N,CHAR)

- a) Z is the mode set array.
- b) X,Y is the location of the center of the first character.
- c) N is the number of characters to type ($N \geq 1$).
- d) CHAR is the name of a variable or an array that contains the characters to type.

- 3) To draw joined vectors,

CALL METAZZ(Z,8,N,X,Y,Q,Q)

- a) Z is the mode set array.
- b) N is the number of x,y coordinates. N - 1 joined line segments will be drawn.
- c) X,Y are arrays of x,y coordinates. A line is drawn between each point in the arrays.

4) To draw line segments,

CALL METAZZ(Z,9,N,X1,Y1,X2,Y2)

- a) Z is the mode set array.
- b) N is the number of line segments to draw. If N > 1, then X1,Y1 and X2,Y2 must be arrays of points.
- c) X1,Y1 are arrays containing the starting points of each line.
- d) X2,Y2 are arrays containing the terminal points of each line.

Modifications

All mode sets are in effect when METAZZ is called.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

PACKZZ: Packs Bytes into the Output Buffer

PURPOSE

Packs bytes into a buffer, then writes the filled buffer onto an output device. PACKZZ is used by SNAPG to produce 2250 hardcopy on the S-C 4060.

USAGE

Standard Calls

CALL PACKZZ(Z,I,IDATA,NTH,NO)

- 1) Z is the mode set array.
- 2) I tells whether IDATA contains characters.
If I = 0, no characters.
= 3, IDATA contains characters. This distinction allows character conversion within PACKZZ.
- 3) IDATA is a word or an array containing the byte string to pack into the buffer.
- 4) NTH specifies at which byte in the string to begin (NTH \geq 1).
- 5) NO specifies the number of bytes of data to pack.

Special Calls

- 1) To open the output buffer,
CALL PACKZZ(Z,-1,0,0,0)
- 2) To close the buffer,
CALL PACKZZ(Z,1,0,0,0)
- 3) To flush the output buffer without closing it,
CALL PACKZZ(Z,2,0,0,0)

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

PZZ: Prints Out the Mode Set Array

PURPOSE

Prints the contents of the mode set array. Debugging is PZZ's primary function.

USAGE

Standard Calls

CALL PZZ(Z)

Z is the mode set array.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

2

REPKZZ: Repacks the 2250 Buffer

PURPOSE

Automatically repacks the full 2250 output buffer.
All deleted displays are overlaid at this time.

USAGE

Standard Calls

CALL REPKZZ(\bar{Z})

Z is the mode set array.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

SCALZZ: Converts User Coordinates to Rasters

PURPOSE

Scales user coordinates and converts them to integer rasters.

USAGE

Standard Calls

CALL SCALZZ(Z,X,Y,IX,IY)

- 1) Z is the mode set array.
- 2) X,Y are the x,y coordinates to scale.
- 3) IX,IY contain the rasters on return.

Special Calls

Two functions do the nonlinear scaling. The functions provided in the system do a log transformation to the base 10. The user may substitute any functions of his own to do other transformations. The form of the functions is as follows:

X = XMODZ(X)
Y = YMODZ(Y)

Modifications

Use mode sets to modify SCALZZ as follows:

- 1) To specify the scaling to be done,

CALL SETSMG(Z,14,SCALE)

SCALE specifies the scaling to be done.

SCALE = 0.0 is the default value.

If SCALE = 0.0, scale real subject space coordinates.

- = 1.0, scale real absolute raster units.
- = 2.0, scale integer absolute raster units.
- = 3.0, scale real normalized object space.

2) To specify nonlinear scaling in the x-axis,

CALL SETSMG(Z,23,1.)

To reset the mode to its default value for linear scaling,

CALL SETSMG(Z,23,0.)

3) To specify nonlinear scaling in the y-axis,

CALL SETSMG(Z,24,1.)

To reset the mode to its default value for linear scaling,

CALL SETSMG(Z,24,0.)

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

TABLZZ: Resident Data Section

PURPOSE

Though actually a control section, TABLZZ contains all resident tables, IOBs, CCWs, DCBs, DECBs, etc., needed by IGS. It is the only portion of the graphics system that must remain in core.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language. DCBZZ is an entry to the 2250 DCB.

TRAPZZ: Processes 2250 Input Traps

PURPOSE

Entered asynchronously whenever a light pen, keyboard, function key, or end order sequence interrupt occurs.

USAGE

Standard Calls

OS/360 gives control to TRAPZZ.

MISCELLANEOUS INFORMATION

Written in OS/360 assembly language.

UNSCZZ: Converts Rasters to User Coordinates

PURPOSE

Converts rasters into user coordinates.

USAGE

Standard Calls

CALL UNSCZZ(Z,X,Y,IX,IY)

- 1) Z is the mode set array.
- 2) X,Y contain the user coordinates on return.
- 3) IX,IY are the raster coordinates.

Modifications

Use mode sets to modify UNSCZZ as follows: To specify the scaling to be done,

CALL SETSMG(Z,14,SCALE)

SCALE specifies the scaling to be done. SCALE = 0.0 is the default value.

- If SCALE = 0.0, scale real subject space coordinates.
= 1.0, scale real absolute raster units.
= 2.0, scale integer absolute raster units.
= 3.0, scale real normalized object space.

MISCELLANEOUS INFORMATION

Written in FORTRAN IV.

REFERENCES

1. Davis, M. R., and T. O. Ellis, "The RAND Tablet: A Man-Machine Graphical Communication Device," *AFIPS Conference Proceedings* (1964 FJCC), Vol. 26, Part 1, Spartan Books, Inc., Baltimore, Maryland, 1964, pp. 325-331; also, The RAND Corporation, RM-4122-ARPA, August 1964.
2. "IBM System/360 Component Description: IBM 2250 Display Unit Model 1," Form A27-2701-0, IBM Corporation, Kingston, New York.
3. Groner, G. F., *Real-Time Recognition of Handprinted Text*, The RAND Corporation, RM-5016-ARPA, October 1966.
4. Groner, G. F., *Real-Time Recognition of Handprinted Text: Program Documentation*, The RAND Corporation, RM-5550-ARPA, May 1968.
5. "S-C 4060 Stored Program Recording System: Description and Specifications," Document 9500209, Stromberg-Carlson, San Diego, California, October 1966.
6. "American Standard FORTRAN," American Standards Association, New York, March 7, 1966.
7. "Graphic Specifications for Standard Graphic Subroutines," SHARE Standard Graphic Output Language Committee, August 8, 1966.
8. "IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit," Program Number 360S-10-523, Form C27-6909-4, IBM Corporation, Kingston, New York, 1967.

ALPHABETIC LIST OF SUBROUTINES

ALARMG, p. 94.
AREASG, pp. 88-89.
BUFFZZ, p. 134.
BUTTNG, pp. 90-91.
CBUTZZ, p. 135.
CHANG, p. 95.
CHARG, pp. 80-81.
CHARZZ, pp. 136-138.
CIRARG, pp. 31-32.
CONVTG, p. 96.
CTABZZ, p. 139.
CURSRG, pp. 97-98.
C225ZZ, p. 140.
DATAG, pp. 82-84.
DEBUG, p. 99.
DISPLG, p. 68.
ERRZZ, p. 141
EXITG, p. 29.
FBUTZZ, p. 142.
FMTSG, p. 100.
GETCZZ, p. 143.
GETIDG, p. 22.
GETSMG, p. 27.
GNXYZZ, p. 144.
GPCHRG, pp. 69-70.
GPXYG, pp. 71-72.
GRAPHG, pp. 65-66.
GRIDG, pp. 48-51.
GSTATG, pp. 74-76.
GTIMEG, p. 101.
IFMZZ, pp. 145-146.
LABELG, pp. 52-57.
LEGNDG, pp. 33-35.
LINESG, pp. 36-37.
METAZZ, pp. 147-148.
MLTPLG, pp. 38-39.
MODESG, p. 21.
NUMBRG, pp. 40-42.
OBJCTG, pp. 24-25.
PACKZZ, pp. 149-150.
POINTG, pp. 43-44.
PUSHG, p. 92.
PUTCZZ, p. 143.
PZZ, p. 151.
RECOG, pp. 85-86.
REPKZZ, p. 152.
RSETMG, p. 28.
SCALZZ, pp. 153-154.
SEGMTG, p. 45.
SETFKG, pp. 102-103.
SETUPG, pp. 62-64.
SETSMG, p. 26.
SNAPG, p. 104.
SUBJEG, p. 23.
TABLZZ, p. 155.
TEXTG, p. 46.
TITLEG, pp. 58-61.
TRAPZZ, p. 156.
UNSCZZ, p. 157.
WAITSG, pp. 77-79.
WTIMEG, p. 105.
XMODZ, p. 153.
XNORMZ, p. 24.
YMODZ, p. 153.
YNORMZ, p. 24.

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY THE RAND CORPORATION		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP	
3. REPORT TITLE THE INTEGRATED GRAPHICS SYSTEM FOR THE IBM 2250			
4. AUTHOR(S) (Last name, first name, initial) Brown, Gary D. and Charles H. Bush			
5. REPORT DATE October 1968		6a. TOTAL No. OF PAGES 175	6b. No. OF REFS. 8
7. CONTRACT OR GRANT No. DAHC15 67 C 0141		8. ORIGINATOR'S REPORT No. RM-5531-ARPA	
9a. AVAILABILITY / LIMITATION NOTICES DDC-1		9b. SPONSORING AGENCY Advanced Research Projects Agency	
10. ABSTRACT A programmer's manual for IGS, the Rand interactive computer graphic software, as implemented on the IBM 2250 graphic hardware. Programmed in FORTRAN IV with special facility for addressing a single character directly, IGS is machine-independent and callable from many languages acceptable to an IBM 360, including PL/I. The 360 can only start or stop the 2250 and read or write into it. The user controls the system through the 2250's on-line typewriter, light pen, or function keys; or by writing or drawing on the Rand Tablet, if included; or by touching user-defined sensitive areas with the light pen or Tablet stylus. Only parameters of interest need to be specified; IGS automatically provides default values for character size, spacing, line spacing, and other format details. IGS accepts user coordinates and translates them into raster units. It automatically rescales graphical data to fit available space, converts graphical to numerical data, etc. One subroutine draws an entire graph to fit given data, with a single call. The RM includes an alphabetical index, a cross-reference index of subroutine calls, and a listing by type, as well as a detailed description of each. /)		11. KEY WORDS Computer graphics Computer programming language	