M T S

The Michigan Terminal System

Volume 17: Integrated Graphics System

December 1980

Updated March 1983 (Update 1)
| Updated September 1984 (Update 2)

The University of Michigan Computing Center
Ann Arbor, Michigan

DISCLAIMER

The MTS Manual is intended to represent the current state of the Michigan Terminal System (MTS), but because the system is constantly being developed, extended, and refined, sections of this volume will become obsolete. The user should refer to the Computing Center Newsletter, Computing Center Memos, and future Updates to this volume for the latest information about changes to MTS.

PREFACE

The software developed by the Computing Center staff for the operation of the high-speed processor computer can be described as a multiprogramming supervisor that handles a number of resident, reentrant programs. Among them is a large subsystem, called MTS (Michigan Terminal System), for command interpretation, execution control, file management, and accounting maintenance. Most users interact with the computer's resources through MTS.

The MTS Manual is a series of volumes that describe in detail the facilities provided by the Michigan Terminal System. Administrative policies of the Computing Center and the physical facilities provided are described in a separate publication entitled Introduction to Computing Center Services.

The MTS volumes now in print are listed below. The date indicates the most recent edition of each volume; however, since volumes are periodically updated, users should check the file *CCPUBLICATIONS, or watch for announcements in the Computing Center Newsletter, to ensure that their MTS volumes are fully up to date.

| Volume 1: The Michigan Terminal System, January 1984
| Volume 2: Public File Descriptions, April 1982
| Volume 3: System Subroutine Descriptions, April 1981
| Volume 4: Terminals and Networks in MTS, March 1984
| Volume 5: System Services, May 1983
| Volume 6: FORTRAN in MTS, October 1983
| Volume 7: PL/I in MTS, September 1982
| Volume 8: LISP and SLIP in MTS, June 1976
| Volume 9: SNOBOL4 in MTS, September 1975
| Volume 10: BASIC in MTS, December 1980
| Volume 11: Plot Description System, August 1978
| Volume 12: PIL/2 in MTS, December 1974
| Volume 13: The Symbolic Debugging System, November 1980
| Volume 14: 360/370 Assemblers in MTS, May 1983
| Volume 15: FORMAT and TEXT360, April 1977
| Volume 16: ALGOL W in MTS, September 1980
| Volume 17: Integrated Graphics System, December 1980
| Volume 18: The MTS File Editor, September 1982
| Volume 19: Tapes and Floppy Disks, February 1983

Other volumes are in preparation. The numerical order of the volumes does not necessarily reflect the chronological order of their appearance; however, in general, the higher the number, the more specialized the volume. Volume 1, for example, introduces the user to

MTS and describes in general the MTS operating system, while Volume 10 deals exclusively with BASIC.

The attempt to make each volume complete in itself and reasonably independent of others in the series naturally results in a certain amount of repetition. Public file descriptions, for example, may appear in more than one volume. However, this arrangement permits the user to buy only those volumes that serve his or her immediate needs.

Richard A. Salisbury

General Editor

December 1980

PREFACE TO VOLUME 17

The first part of Volume 17 (everything up to the appendices) presents an overview of the basic uses of the Integrated Graphics (IG) System. As a prerequisite for understanding this material, the reader should have some familiarity with programming in the MTS context. Furthermore, since most of the examples are written in FORTRAN, the reader should have some understanding of this language. No prior knowledge of computer graphics is assumed.

The appendices present specialized material, which is intended more for reference than for straight-through reading.

Appendix I presents descriptions of the use of various I/O devices in IG. However, these descriptions do not extend to the general use of these devices in MTS; for a general description of the use of terminals, the reader should refer to MTS Volume 4, Terminals and Networks in MTS. When using certain terminals or remote plotters, the reader may also want to refer to the manufacturers' operating manuals.

James Blinn, Andy Goodrich, and Kalle Nemvalts wrote the major portion of this volume. Steve Burling provided much of the material in Appendix D. Daniel J. Fox of the Statistical Research Laboratory provided the Great Lakes maps in the section "Device-Dependent Operations". Many other people, both at the Computing Center and in other university units, read drafts of this volume and provided suggestions and corrections.

December 1980

Contents

Page Revised September 1984 December 1980

INTRODUCTION

The Integrated Graphics (IG) system is a library of FORTRAN-callable subroutines for performing graphics operations. The IG system consists of two sections, a device-independent section and a device-dependent section containing a device-dependent routine (DDR) for each supported device type. User programs interact primarily with the device-independent section (see Figure 1).

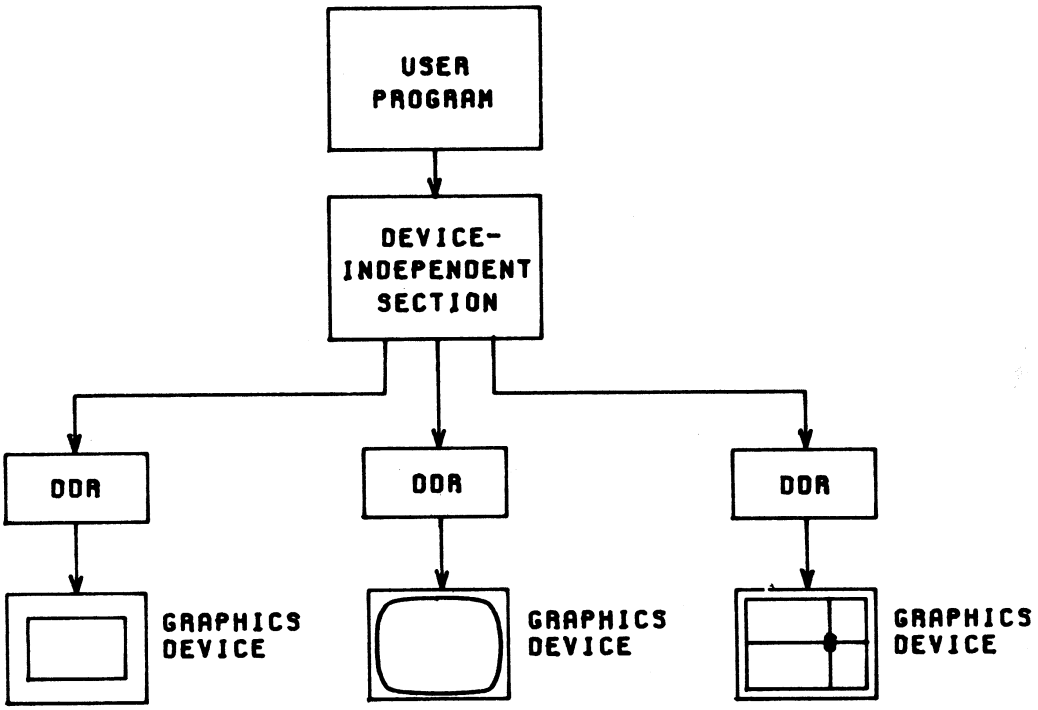


Figure 1: IG System Structure

A program using IG can be run without alteration on a wide variety of graphics terminals. The IG system will automatically take advantage of the extra capabilities of the more sophisticated display devices while simulating these capabilities in software for the simpler (and cheaper) devices.

VALID OUTPUT DEVICES

The IG system supports five major types of graphics terminals (and other output devices):

(1) Standard Storage-Tube-Type Terminals:

Tektronix 4002
Tektronix 4006
Tektronix 4010 Series (e.g., 4010, 4012, 4014)
Computek 400

(2) Terminals with Selective-Erase Capability:

Princeton Electronics Products 801
Magnavox 12000

(3) Terminals with Vector-Refresh-Type Displays:

DEC GT40

(4) Terminals with Raster-Refresh-Type Displays:

Ramtek 6200A

(5) Miscellaneous:

CalComp Plotter
Remote Plotters (e.g., Hewlett-Packard 7203A, Tektronix 4662)
Hard-Copy Terminals with Plot Mode (e.g., DTC300)
IBM-3270-Compatible Terminals
Nongraphics Terminals (e.g., DECwriter, Teletype)
Line Printer

Before IG sends any output to a terminal, it identifies the terminal type, erases the screen, and dynamically loads the appropriate DDR. Most terminals are identified automatically. However, when using a | terminal with no automatic answerback facility, the user must specify | the terminal type to the network by using the TERMINAL device command. | To use these terminals, the user should follow the procedures described in Appendix I.

New types of graphics terminals can be added to the IG system by constructing the appropriate DDRs. Users with new types of graphics terminals are invited to contact the Computing Center for information about how to add their particular device to the system.

BASIC USE OF THE IG SYSTEM

The IG system can be used for plotting both two- and three-dimensional data. Some typical products are graphs, charts, maps, signs, technical illustrations, perspective drawings, etc.

This section covers the basic procedure for displaying a two-dimensional picture on a terminal. The various steps in this procedure are covered in more detail in later sections. Appendix B provides more detailed information on each subroutine in the IG system.

All examples are written in FORTRAN. However, other languages can just as easily be used, provided that S-type calling conventions are observed for all IG subroutine calls.

RUNNING A PROGRAM THAT USES IG

To use the IG system, the user must provide a program that makes calls to the IG subroutines. When executing such a program, the user must concatenate the public file *IG to the object file of the program:

```
$RUN objectfile+*IG
```

INTERFACE WITH THE PLOT DESCRIPTION SYSTEM

The Plot Description System (PDS) is a subroutine package for plotting data on the CalComp plotter (see MTS Volume 11, Plot Description System). PDS resides in the public file *PLOTSYS. PDS includes subroutines which generate coordinate axes, grids, line graphs, bar graphs, etc. A program which uses PDS may also use the interactive facilities of IG, if it is executed as follows:

```
$RUN objectfile+*IG+*PLOTSYS
```

The plots produced by PDS will be intercepted by IG and displayed on the terminal screen instead of being written into a PDS file. See Appendix J for further details.

INITIALIZING THE IG SYSTEM

Before calling any other IG subroutine, the user program must call IGINIT to initialize the internal IG data structure:

```
CALL IGINIT
```

BASIC IG CONCEPTS

The IG system is designed to isolate the user from the details of how a particular terminal generates or alters the screen image. In keeping with this philosophy, the IG data structure is device-independent. In IG, the basic data unit is the picture. A picture consists of a set of line segments in a two- or three-dimensional coordinate space. Each line segment is represented by the coordinates of its endpoints; these may assume any floating-point values. Pictures are generated by the user program via calls to the IG picture-description subroutines.

To display pictures on the terminal screen, the user program may call the subroutine IGDRON. This subroutine maps pictures into screen coordinates for viewing (see Figure 2). The screen coordinates are defined in such a way that the origin (0,0) corresponds to the center of the screen, the point (+1,0) corresponds to the extreme right edge of the screen, and the point (-1,0) corresponds to the extreme left edge of the screen. The Y coordinates (vertical axis) have the same units as the X coordinates (horizontal axis). Thus, if the particular terminal has a rectangular screen, the maximum Y coordinate will be less than 1. The value of the maximum Y coordinate is called the aspect ratio of the screen (the ratio of the vertical dimension to the horizontal dimension).[1]

By default, pictures are mapped onto the screen in such a way that each picture coordinate is mapped into the screen coordinate of the same value. Picture coordinates may assume any floating-point values, but when a picture is mapped onto the screen, any lines extending off the screen are clipped off at the edges. (This clipping affects only the screen image, not the picture itself.)

---

[1]The screen coordinates may be redefined so that they occupy the maximum square area centered within the screen, i.e., so that the X and Y coordinates both have values between -1 and +1. This can easily be done by calling the subroutine IGCTRL (see the section "Device-Dependent Operations").

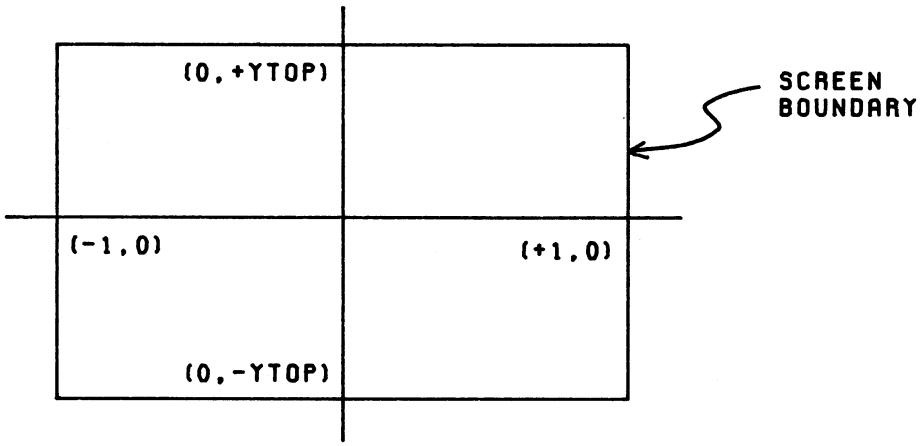


Figure 2: Screen Coordinates

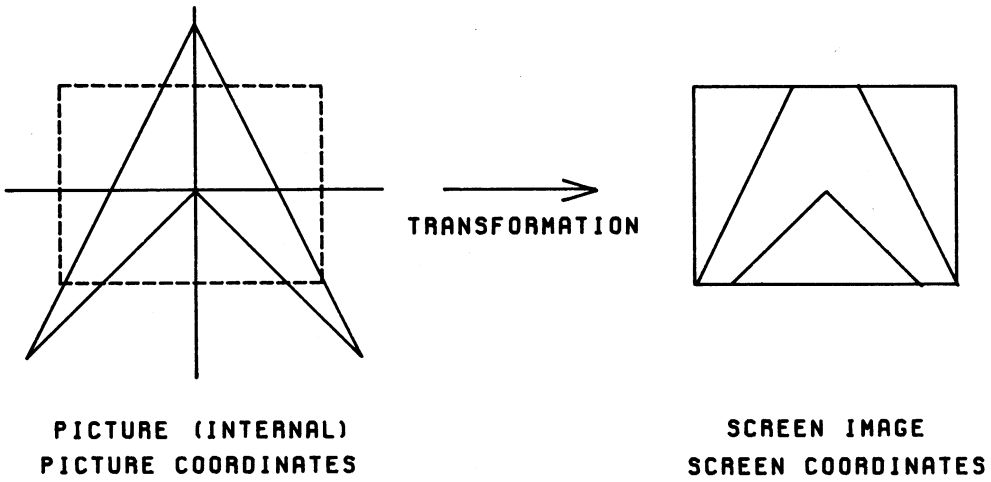


Figure 3: Displaying a Picture on the Screen

A transformation will generally be applied to a picture when it is mapped onto the screen (see Figure 3). For example, the picture may be scaled (magnified or reduced), translated (shifted sideways or up or down), or rotated about the origin. Following the transformation, any lines extending off the screen will be clipped off at the edges. Transformations are specified by the user program via IG subroutine calls.

In summary, a picture is an internal entity and is defined in device-independent coordinates. When the picture is mapped onto the screen, it will generally be transformed in some appropriate way (e.g., the picture might be reduced so that it will fit within the screen boundaries). The IG system will automatically generate any hardware commands necessary to generate the actual screen image.

GENERATING A PICTURE

Beginning a Picture

To begin a new picture, the user program must call the subroutine IGBGNS:

```
CALL IGBGNS(NAMPIC)
```

NAMPIC must contain a four-character picture name which must begin with a letter. In FORTRAN, this may conveniently be specified as a literal string:

```
CALL IGBGNS('PICT')
```

If the literal string contains more than four characters, only the first four will be used.

A call to IGBGNS establishes the named picture as the active picture, i.e., the picture to which lines and text are to be added. If the named picture already exists, its previous contents are discarded.

"Drawing" Lines

Within the picture coordinate space, there is a pointer that may be thought of as a "pen" that may be moved around to "draw" the lines in the picture. (Remember that the picture is internal; nothing is visible on the screen until the picture is mapped onto the screen.) This pen may be in either the down or up position. When the pen is moved in the down position, a visible line is produced; when the pen is moved in the up position, an invisible line is produced. IG always remembers the

current position of the pen, i.e., the coordinates of the pen within the picture coordinate space. Following a call to IGBGNS, the initial current position is the origin (0,0).

Lines may be added to the active picture by calling any combination of the following four subroutines:

CALL IGMA(X,Y)

Move (draw an invisible line) from the current position to the absolute position (X,Y).

CALL IGDA(X,Y)

Draw a visible line from the current position to the absolute position (X,Y).

CALL IGMR(DX,DY)

Move with a displacement (DX,DY) relative to the current position.

CALL IGDR(DX,DY)

Draw with a displacement (DX,DY) relative to the current position.

Adding Text

A text string may be added to the active picture by calling the following subroutine:

CALL IGTXT('textstring<E>')

The string may contain up to 256 characters, terminated by the control operand <E>. When the string is placed in the picture coordinate space, the lower-left corner of the first character is placed at the current position (as determined by the last call to IGMA, etc.). The default character size will be the hardware character size of the terminal. Most terminals have a hardware character size of about .027 in the screen coordinate system. If a terminal has more than one hardware character size, the size closest to .027 is used. After the string has been placed in the picture coordinate space, the current position will be at the lower-right corner of the last character.

Terminating a Picture

When all of the lines and text in a picture have been specified, the user program may call IGENDS to terminate the picture:

```
CALL IGENDS(NAMPIC)
```

NAMPIC must contain the same name that was used with IGBGNS. Following the call to IGENDS, NAMPIC is no longer the active picture.

TRANSFORMING A PICTURE

Associated with each picture is a transformation, which is applied when the picture is mapped onto the screen. By default, this transformation is the identity transformation, i.e., each picture coordinate is mapped into the screen coordinate of the same value. However, the user program may at any time call the subroutine IGTRAN to specify a nontrivial transformation. For example,

```
CALL IGTRAN(NAMPIC,'MOVE',-XCENTR,-YCENTR,'SCALE',FACTOR)
```

can be used to zoom in on a region centered on (XCENTR,YCENTR). Here,

```
NAMPIC is the name of the picture as passed to IGBGNS
XCENTR is the X coordinate of the center of the region
YCENTR is the Y coordinate of the center of the region
FACTOR is the scale factor
```

This transformation will be applied the next time the picture is displayed. The picture will be moved horizontally by a displacement -XCENTR and vertically by a displacement -YCENTR, and then scaled by FACTOR. For example,

```
CALL IGTRAN('PICT','MOVE',-.5,-.5,'SCALE',2.)
```

will move the picture left by 0.5 and down by 0.5, and then double its size.

Recall, once again, that a picture is an internal entity, independent of the current screen image. The user program can draw the picture (via IGMA, IGDA, etc.) in any coordinates appropriate to the problem at hand, and then specify a transformation such that the picture will fit within the screen boundaries. A transformation does not alter the various line segments that make up the picture; rather, it determines the way that the picture will be mapped onto the screen. The same picture can be transformed and displayed in several different ways.

DISPLAYING PICTURES

The user program may at any time call IGDRON to display pictures on the screen of the terminal:

```
CALL IGDRON('TERMINAL')
```

IG will erase the screen, apply the transformation associated with each picture, and generate any hardware commands necessary to generate the screen image. (If several pictures have been created, they will all be displayed, superimposed on one another.) Alternatively, the user program may call IGDRON to generate hard copy that corresponds to the screen image:

```
CALL IGDRON('CALCOMP')
```

IG will write a plot description onto logical I/O unit 9, which should be assigned to a file or other storage device. This plot description may be used to generate a real CalComp plot (see Appendix K). The plot will be scaled to fit into an 8.5x11-inch sheet of plotter paper. The -1 to +1 visible region will be mapped into a 7.5x7.5-inch square with an X margin of 0.5 inches and a Y margin of 1.75 inches. Thus, a line of length 1.0 in the screen coordinate system will be 3.75 inches long on the plot.

A typical user program may call IGDRON('TERMINAL'), ask the user if hard copy is desired, and depending on the reply, call IGDRON('CALCOMP').

MODIFYING A PICTURE

The contents of a picture may be replaced in the following way:

```
CALL IGBGNS(NAMPIC)
CALL IGMA(X1,Y1)     ┐
CALL IGDA(X2,Y2)     | calls to specify
       ...           ┘ new lines and text
CALL IGENDS(NAMPIC)
```

A call to IGBGNS with the name of an existing picture "empties" it of all previous lines and text, and makes it the active picture. Then a series of calls to picture-description subroutines specifies the new contents of the picture. Finally, a call to IGENDS terminates the picture. Upon the next call to IGDRON('TERMINAL'), the screen will be erased and the new picture displayed.

An existing, nonactive picture can be continued in the following way:

```
CALL IGCTNS(NAMPIC)
CALL IGMA(X1,Y1)     ┐
CALL IGDA(X2,Y2)     │ calls to add
       ...           ┘ new lines and text
CALL IGENDS(NAMPIC)
```

A call to IGCTNS with the name of an existing picture makes it the active picture, but does not alter its contents. The current position remains the same as when the picture was last active. Following the call to IGCTNS, a series of calls to picture-description subroutines adds lines and text to the picture. Finally, a call to IGENDS terminates the picture.

The lines and text of a picture are stored separately from the transformation associated with the picture. Thus, a call to IGBGNS or IGCTNS will not affect any transformation previously specified by IGTRAN. The existing transformation will be applied to the redefined picture.

PICTURE-DESCRIPTION SUBROUTINES

This section covers in more detail the use of the IG picture-description subroutines. These subroutines add lines and text to the active picture. Once the picture has been generated, IGDRON may be called to display it on the screen.

LINES

Single Lines

Lines may be added to the active picture by calling any combination of the following four subroutines:

CALL IGMA(X,Y)

Move (draw an invisible line) from the current position to the absolute position (X,Y).

CALL IGDA(X,Y)

Draw a visible line from the current position to the absolute position (X,Y).

CALL IGMR(DX,DY)

Move with a displacement (DX,DY) relative to the current position.

CALL IGDR(DX,DY)

Draw with a displacement (DX,DY) relative to the current position.

Multiple Lines

Repeated calls to the above single-line subroutines can be inefficient and inconvenient. The subroutine IGVEC accepts an entire vector of X and Y coordinates and performs a sequence of moves and draws with one call. The simplest call to this subroutine is:

```
CALL IGVEC(N,X,Y)
```

Here, X and Y are REAL*4 vectors with at least N elements. This call will perform a move to the absolute coordinate (X(1),Y(1)) followed by draws to the absolute coordinates (X(2),Y(2)) through (X(N),Y(N)).

The draw/move (visible/invisible) status of each line may be explicitly specified by the optional INTVEC parameter:

```
CALL IGVEC(N,INTVEC,X,Y)
```

The INTVEC parameter may be given in one of two ways. The first way is to make it an INTEGER*4 vector of length N. A value of INTVEC(I)=0 means that the Ith line is to be a move. A value of INTVEC(I)>0 means that the Ith line is to be a draw. Thus, the result of the calls

```
CALL IGMA(-.5,+.5)
CALL IGDA(+.5,+.5)
CALL IGMA(-.5,-.5)
CALL IGDA(+.5,-.5)
```

could also be achieved by:

```
INTEGER*4 INTVEC(4)/  0,  1,  0,  1/
REAL*4 X(4)        /-.5,+.5,-.5,+.5/
REAL*4 Y(4)        /+.5,+.5,-.5,-.5/
CALL IGVEC(4,INTVEC,X,Y)
```

The second way to give the INTVEC parameter is to make it a literal string. Some of the permissible literal string values are:

'D '	All lines are draws. (Draw,Draw,...)
'M(D)'	First line is a move and the rest are draws. (Move,Draw,Draw,Draw,...)
'MD '	Odd lines are moves, while even lines are draws. (Move,Draw,Move,Draw,...)

The literal string is interpreted as follows. The character "M" or "D" indicates a move or a draw for the current line. A " " (blank) indicates that the scanning of the string is to be resumed from the beginning. A ")" indicates that scanning is to be resumed from the last previous "(". Thus, the above example could also be performed by

```
CALL IGVEC(4,'MDMD',X,Y)
```

or

```
CALL IGVEC(4,'(MD)',X,Y)
```

If the INTVEC parameter is omitted, it is assumed to have the value 'M(D)', i.e., the first line is a move and all subsequent lines are draws.

The relative/absolute status of each line may be explicitly specified by the optional MOVTYP parameter:

```
CALL IGVEC(N,INTVEC,MOVTYP,X,Y)
```

The MOVTYP parameter may be given only if the INTVEC parameter is also given. The MOVTYP parameter may be given in one of two ways. The first way is to make it an INTEGER*4 vector of length N. A value of MOVTYP(I)=0 means that the Ith line is to be drawn relative to the (I-1)th. A value of MOVTYP(I)>0 means that the Ith line is to be drawn absolute. Thus, the result of the calls

```
CALL IGMA(+.5,+.5)
CALL IGDR(-.1,+.1)
CALL IGDR(+.1,+.1)
```

could also be achieved by:

```
INTEGER*4 MOVTYP(3)/  1,  0,  0/
REAL*4 X(3)         /+.5,-.1,+.1/
REAL*4 Y(3)         /+.5,+.1,+.1/
CALL IGVEC(3,'MDD',MOVTYP,X,Y)
```

The second way to give the MOVTYP parameter is to make it a literal string. Some of the permissible literal string values are:

'A '	All lines are absolute. (Abs,Abs,...)
'R '	All lines are relative. (Rel,Rel,...)
'(AR)'	Odd lines are absolute, while even lines are relative. (Abs,Rel,Abs,Rel,...)

The literal string is interpreted in the same way as with the INTVEC parameter. Thus, the result of the above example could also be achieved by:

```
CALL IGVEC(3,'MDD','A(R)',X,Y)
```

If the MOVTYP parameter is omitted, it is assumed to have the value '(A)', i.e., all lines are absolute moves or draws.

Some of the elements in the X and Y vectors may be skipped by the IGVEC subroutine, depending on the value of the optional NWORDS parameter:

```
CALL IGVEC(N,INTVEC,MOVTYP,NWORDS,X,Y)
```

Here, NWORDS is the integer number of fullwords separating the elements to be plotted. As an example, if NWORDS=3, the elements to be plotted are (X(1),Y(1)), (X(4),Y(4)), (X(7),Y(7)), etc. As another example, if NWORDS=2, the elements to be plotted are (X(1),Y(1)), (X(3),Y(3)), etc. The NWORDS parameter may be given only if both the INTVEC and MOVTYP parameters are also given. If the NWORDS parameter is omitted, it is assumed to have the value 1.

TEXT

IG uses two different methods to display text. The first way is to generate the characters in software as a series of line segments. The second way is to use the hardware character set of the terminal (i.e., the keyboard character set that is used for nongraphic I/O operations).

Software-generated characters are drawn inside imaginary rectangular envelopes which include space to separate adjacent characters. The envelope height is the text scale. In general, the envelope width may vary from character to character. For the default IG character set, however, the envelope shape is always square (see Figure 4). Software-generated text may have any scale. When starting a text string, IG places the lower-left corner of the first character at the current position. In the process of drawing the string, IG moves the current position to the lower-right corner of the last character. This kind of text can be transformed along with the rest of the picture.

On most terminals, hardware-generated characters are displayed as dot matrices inside rectangular envelopes (see Figure 5). The exact manner in which this is done depends on the kind of terminal. The envelope width is the text scale, which is approximately .027 in the -1 to +1 screen coordinate system (depending on the kind of terminal). If the terminal has multiple text scales, the one closest to .027 is used. When starting a text string, IG places the lower-left corner of the first character at the current position. In the process of inserting the string, IG moves the current position to the lower-right corner of the last character. This kind of text can be transformed only in a limited way: the position of the text (i.e., the lower-left corner of the first character) can be transformed along with the rest of the picture, but the text itself cannot be scaled or rotated.

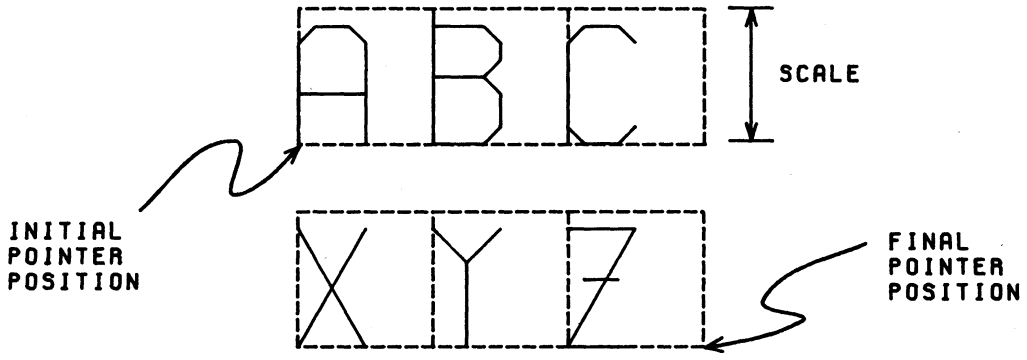


Figure 4: Text Generated in Software

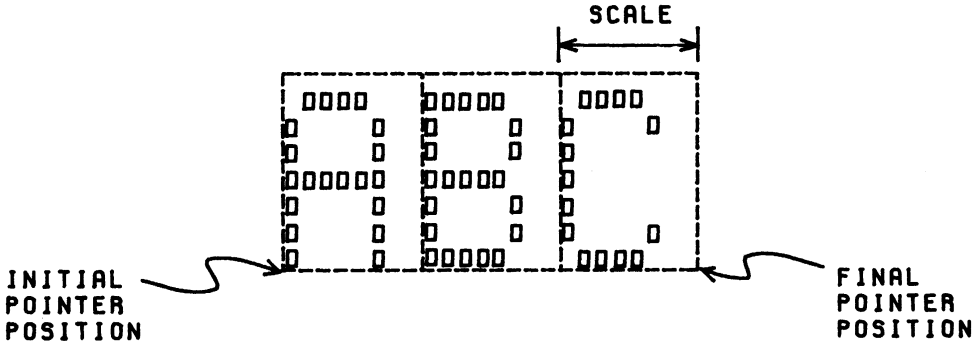


Figure 5: Text Generated by Hardware

There are five subroutines for adding text to the active picture. The following list describes the kind of text added by each one.

IGTXT — Constant text string, subject to scaling and rotation of the picture, generated by either software or hardware.

IGTXTH — Constant text string, generated by hardware.

IGFMT — Character representation of a variable converted via a FORTRAN-type format, subject to scaling and rotation of the picture, generated by either software or hardware.

IGFMTH — Character representation of a variable converted via a FORTRAN-type format, generated by hardware.

IGSYM — One-character plotting symbol, centered at the current position, generated by either software or hardware.

After a call to any of the first four subroutines, the current position will be at the lower-right corner of the last character. Thus, calls to IGTXT and IGFMT (or IGTXTH and IGFMTH) can be intermixed to add one line of text.

Constant Text Strings

The subroutines IGTXT and IGTXTH are used to add constant text strings to the active picture. Each of these subroutines takes as a parameter a text string (usually a literal string). This text string may contain up to 256 characters, terminated by the control operand <E>.

Whenever possible, text added by IGTXT will be generated by hardware, at a scale of approximately .027 (depending on the kind of terminal). However, if a rotation or scaling is applied to the picture, the text will be generated in software and will be transformed along with the rest of the picture.

Text added by IGTXTH will be generated by hardware (except when the output device has no hardware character set, in which case the text will be generated in software, at a scale of .027). If a rotation or scaling is applied to the picture, the position of the text (i.e., the lower-left corner of the first character) will be transformed along with the rest of the picture. However, the text itself will not be scaled or rotated.

A text string passed to IGTXT or IGTXTH will include control operands of the form <op>. The following list describes the valid control operands:

<E> indicates the end of the text string. (This control operand is mandatory.) For example,

```
CALL IGTXT('HI THERE<E>')
```

will produce:

HI THERE

The lower-left corner of the H is placed at the current position. Upon exit, the current position will be at the lower-right corner of the second E.

<CRLF> performs a "carriage return", inserting the following part of the text string immediately under the preceding part. For example,

```
CALL IGTXT('ONE<CRLF>TWO<CRLF>THREE<E>')
```

will produce:

ONE
TWO
THREE

<BSUP>,<ESUP> delimit text that is to appear in superscript form. The control operand <BSUP> begins a superscript and the control operand <ESUP> ends a superscript. Superscripts are elevated and made smaller than the preceding text. For example,

```
CALL IGTXT('2<BSUP>5<ESUP>=32<E>')
```

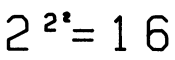
will produce:

$2^5=32$

It is also possible to have superscripted superscripts. These can be produced by nesting the superscript delimiters. For example,

```
CALL IGTXT('2<BSUP>2<BSUP>2<ESUP><ESUP>=16<E>')
```

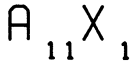
will produce:



<BSUB>,<ESUB> delimit text that is to appear in subscript form. Subscripts are produced in a manner similar to superscripts, but are lowered instead of elevated. For example,

```
CALL IGTXT('A<BSUB>11<ESUB>X<BSUB>1<ESUB><E>')
```

will produce:



It is possible to nest subscript delimiters within superscript delimiters to produce subscripted superscripts. For example,

```
CALL IGTXT('A<BSUP>I<BSUB>5<ESUB><ESUP><E>')
```

will produce:

Format Conversion

The subroutines IGFMT and IGFMTH are used to convert a variable value into character form and add the resulting text string to the active picture. A text string produced by IGFMT will be treated just like a text string produced by IGTXT, and a text string produced by IGFMTH will be treated just like a text string produced by IGTXTH. Calls to IGFMT and IGFMTH take the form

```
CALL IGFMT(VAR,FORMAT[,NCHARS[,NDEC]])

CALL IGFMTH(VAR,FORMAT[,NCHARS[,NDEC]])
```

where:

```
VAR    = the variable to be converted
NCHARS = the number of characters in the output field
NDEC   = the number of digits after the decimal point
FORMAT = a code for the kind of conversion:

         'A'  Character string
         'F'  REAL*4
         'I'  INTEGER*4
         'E'  REAL*4    (exponential notation)
         'D'  REAL*8    (exponential notation)
         'H'  INTEGER*2
```

The FORMAT, NCHARS, and NDEC parameters work in a manner similar to normal FORTRAN formats (except that the H code has a meaning different from its meaning in FORTRAN). If NCHARS is omitted, the output field will be just long enough to contain the converted variable (except when the format is A, in which case the output field will be of length 1). For example,

```
J=25
CALL IGFMT(J,'I',5)   generates   "   25"
CALL IGFMT(J,'I')     generates   "25"

A=3.25
CALL IGFMT(A,'F',7,2) generates   "   3.25"
CALL IGFMT(A,'F')     generates   "3.25"
```

Intermixing Text Subroutine Calls

After a call to IGTXT or IGFMT, the current position will be at the lower-right corner of the last character. Thus, calls to IGTXT and IGFMT (or IGTXTH and IGFMTH) may be intermixed to produce single lines of text. For example,

```
X = 2.57
CALL IGTXT('X=<E>')
CALL IGFMT(X,'F')     generates   "X=2.57"
```

Local Scaling of Text

Whenever possible, text added by IGTXT or IGFMT will be generated by hardware, at a scale of approximately .027 (depending on the kind of terminal). This scale may be changed by a call to IGTXT, in which case the text will be generated in software. A call of the form

```
CALL IGTXT('<ASCL>',SCALE)
```

specifies an absolute scale (in picture coordinates) for subsequent characters. This scale is given as the second parameter. For example,

```
CALL IGTXT('<ASCL>',0.5)
```

specifies an absolute scale of 0.5 for subsequent characters. On the other hand, a call of the form

```
CALL IGTXT('<RSCL>',FACTOR)
```

specifies a relative scale for subsequent characters. The new scale is obtained by multiplying the previous scale by FACTOR. For example,

```
CALL IGTXT('<RSCL>',1.5)
```

specifies that subsequent characters are to be 1.5 times as large as the previous characters. A call to IGTXT may include a combination of scale specifications and text. For example,

```
CALL IGTXT('<RSCL>',3.0,'BIG LETTERS<E>')
```

triples the current scale and then adds the text string "BIG LETTERS" at the new, larger scale. A call to IGTXT may include more than one scale specification. For example,

```
CALL IGTXT('<ASCL>',0.85,'ABC<E>','<RSCL>',4.0,'DEF<E>')
```

adds the text string "ABC" at the absolute scale of 0.85 and then adds the text string "DEF" at a scale 4.0 times as large (an absolute scale of 3.4).

After a scale has been specified by a call to IGTXT, this scale will be used for all subsequent text added by IGTXT and IGFMT, until one of the following conditions occurs:

(1) Another scale is specified by a call to IGTXT.
(2) A call is made to an IG subroutine other than IGTXT or IGFMT.

If (2) occurs, the scale will revert back to its default value (normally the scale of hardware characters).

A default text scale other than the hardware text scale may be specified for a given picture. This default text scale is one of the picture attributes and may be specified by calling IGATTS (see the section "Picture Attributes").

Alternate Character Sets and Fonts

The standard IG character set is listed in Appendix E. This character set is available in hardware on most of the various devices supported by IG. If this character set is not available in hardware, it will be generated in software.

The standard IG character set is also known as the uppercase ASCII character set. Here, the term "uppercase ASCII" refers only to the selection of characters included. (Internally, all characters are represented in EBCDIC codes.)

Alternate character sets are also available. For example, the full 7-bit ASCII character set includes the lowercase letters as well as all of the uppercase ASCII characters.

When using any character set or font, each character is represented by a unique 7-bit ASCII character. For example, when using the 'GREEK.1' character set, the lowercase "alpha" is represented by a lowercase "a". Each character set or font is listed in Appendix E, together with a translation table that relates it to the 7-bit ASCII character set.

Whenever possible, text added by IGTXT, IGFMT, or IGSYM will be generated by hardware, using the uppercase ASCII character set. An alternate character set or font may be specified by a call to IGTXT. (If the character set or font is not available in hardware, it will be generated in software.) A call of the form

```
CALL IGTXT('<FONT>',CSET)
```

specifies a character set or font. The second parameter is a keyword giving the name of the character set or font:

'STANDARD'	uppercase ASCII
'7ASCII'	full 7-bit ASCII
'SANSERIF.1'	
'SANSERIF.2'	
'SANSERIF.CART'	
'ROMAN.2A'	
'ROMAN.2'	
'ROMAN.3'	
'ITALIC.2A'	
'ITALIC.2'	
'ITALIC.3'	

(over)

```
'SCRIPT.1'
'SCRIPT.2'
'GREEK.1'
'GREEK.2A'
'GREEK.2'
'GREEK.CART'
'GREEK'
'GOTHIC.ENGLISH'
'GOTHIC.FRAKTUR'
'GOTHIC.ITALIAN'
'CYRILLIC.2'
```

These keywords must be spelled out in full. (This is an exception to the rule that, with most IG keywords, the first four letters are sufficient.) For example, the following call specifies that the full 7-bit ASCII character set is to be used for subsequent text:

```
CALL IGTXT('<FONT>','7ASCII')
```

A call to IGTXT may include a combination of character set specifications and text. For example,

```
CALL IGTXT('<FONT>','7ASCII','a<E>',
                          '<FONT>','STANDARD','b<E>')
```

adds the text string "a" in lowercase, followed by the text string "B" in uppercase.

After a character set or font has been specified, it will be used for all subsequent text added by IGTXT, IGFMT, and IGSYM, until one of the following conditions occurs:

(1) Another character set or font is specified by a call to IGTXT.
(2) A call is made to an IG subroutine other than IGTXT, IGFMT, or IGSYM.

If (2) occurs, the character set or font will revert back to the default (normally uppercase ASCII).

A default character set or font other than uppercase ASCII may be specified for a given picture. This default character set or font is one of the picture attributes and may be specified by calling IGATTS (see the section "Picture Attributes").

PICTURE MANIPULATIONS

MULTIPLE PICTURES

The user program may create as many pictures as desired. Each picture has its own transformation and may be manipulated separately. When IGDRON('TERMINAL') is called, all existing pictures are displayed on the screen, superimposed on one another.

As an example, suppose the user program generates a set of coordinate axes and then plots a graph. If the axes and the graph are placed in separate pictures, they may be manipulated separately. In particular, the graph may be transformed without affecting the axes (see Figure 6). When IGDRON('TERMINAL') is called, the graph and the axes are superimposed.

Creating Pictures

A picture is created by a call to IGBGNS:

```
CALL IGBGNS(NAMPIC)
       .            ] calls to picture-
       .            ] description subroutines
CALL IGENDS(NAMPIC)
```

NAMPIC must contain a four-character picture name which must begin with a letter. The picture name is usually expressed in the form of a literal string. If the string has more than four characters, only the first four will be used. The picture name will be used to refer to the picture, e.g., for the purpose of specifying transformations. Following the call to IGBGNS, NAMPIC becomes the active picture. Lines and text may be added by making calls to picture-description subroutines. The picture is terminated by a call to IGENDS, giving the same parameter as the call to IGBGNS. This parameter serves as an error check against unmatched IGBGNS...IGENDS pairs.

The user program may also create pictures dynamically, i.e., according to conditions that exist at execution time. Instead of supplying picture names, the user program may call IGBGNS as a function, giving 0 as the parameter:

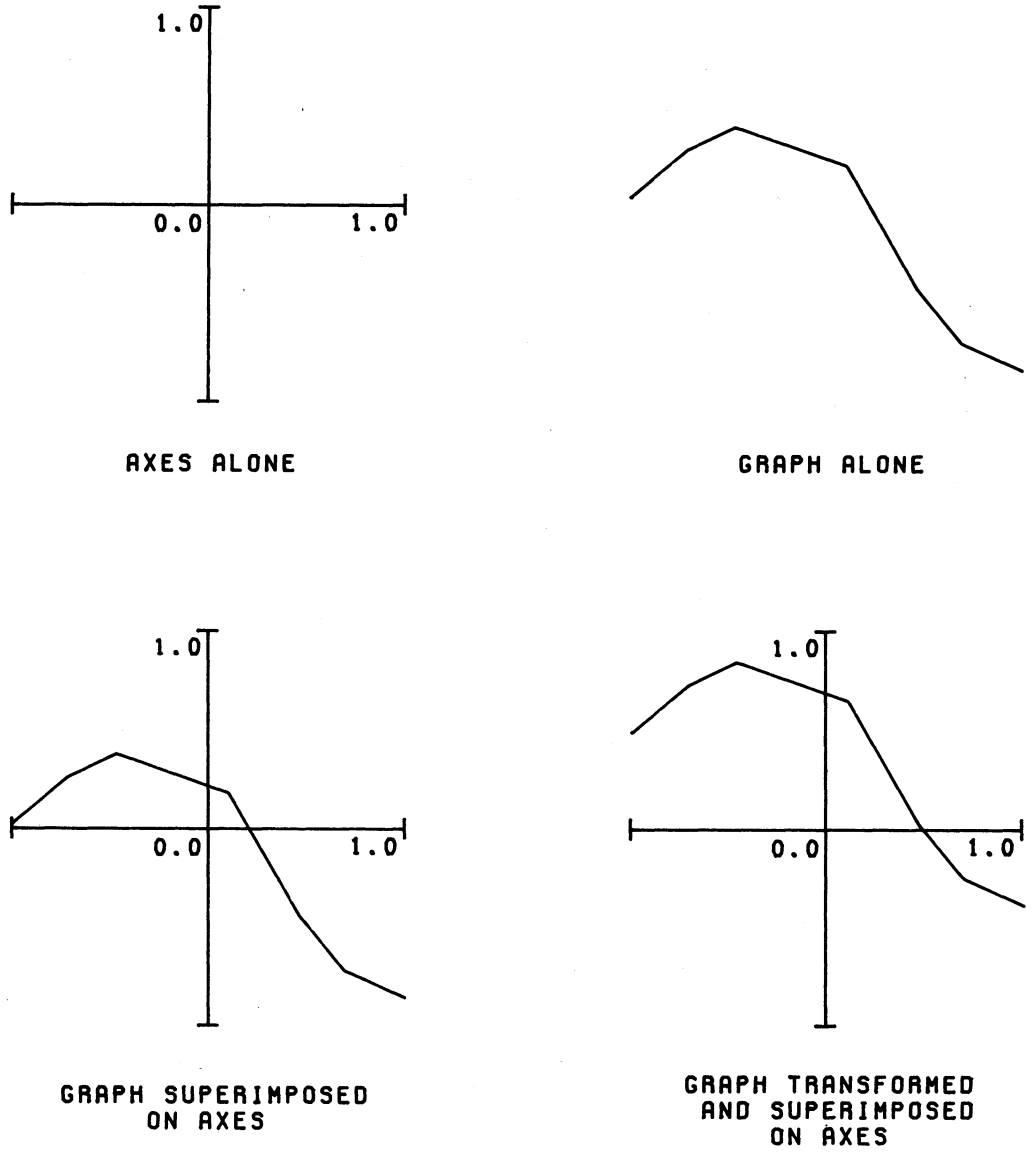


Figure 6: Multiple Pictures

```
INTNAM = IGBGNS(0)
       .            ] calls to picture-
       .            ] description subroutines
CALL IGENDS(INTNAM)
```

IG will generate a unique, internal name for each new picture and return it as the function value of IGBGNS. This name will be some large, positive integer.

Each picture has its own current position. Following a call to IGBGNS, the current position is set to the origin (0,0). The current position may be obtained at any time by calling the IGINFO subroutine. See the IGINFO description in Appendix B.

Nested vs. Nonnested Pictures

A pair of pictures may be nested, i.e., the IGBGNS...IGENDS calls creating one picture may fall within the IGBGNS...IGENDS calls creating the other picture:

```
CALL IGBGNS('PIC1')
       .            ┌──┐
       .            │  │
CALL IGBGNS('PIC2')  │  │
       .            ]  │
       .               │
CALL IGENDS('PIC2')  │  │
       .            │  │
       .            └──┘
CALL IGENDS('PIC1')
```

In this case, 'PIC2' is said to be a subpicture of 'PIC1'. The transformation associated with 'PIC2' maps it not into screen coordinates but into the picture coordinates of 'PIC1'. This situation will be covered in more detail in the section "Subpictures".

For the remainder of this section, all pictures will be assumed to be nonnested, i.e., the IGBGNS...IGENDS pairs will be independent:

```
CALL IGBGNS('PICA')
       .            ]
       .            ]
CALL IGENDS('PICA')
       .
       .
CALL IGBGNS('PICB')
       .            ]
       .            ]
CALL IGENDS('PICB')
```

In this case, each picture transformation maps the picture directly into screen coordinates.

PICTURE TRANSFORMATIONS

Each picture has an associated transformation, which is applied when the picture is mapped onto the screen. By default, this transformation is the identity transformation, i.e., each picture coordinate is mapped into the screen coordinate of the same value. However, the user program may call IGTRAN to specify a nontrivial transformation:

```
CALL IGTRAN(NAMPIC,TYPE,modifs,...)
```

NAMPIC is the name of the picture. TYPE is a four-character keyword (usually given as a literal string) which specifies the transformation type. The following parameters (modifs,...) provide additional data describing the transformation. The transformation is stored with the picture and is used by IGDRON to map the picture into screen coordinates. The transformation is independent of the picture contents (lines and text) and may be specified either before or after the picture contents are specified.

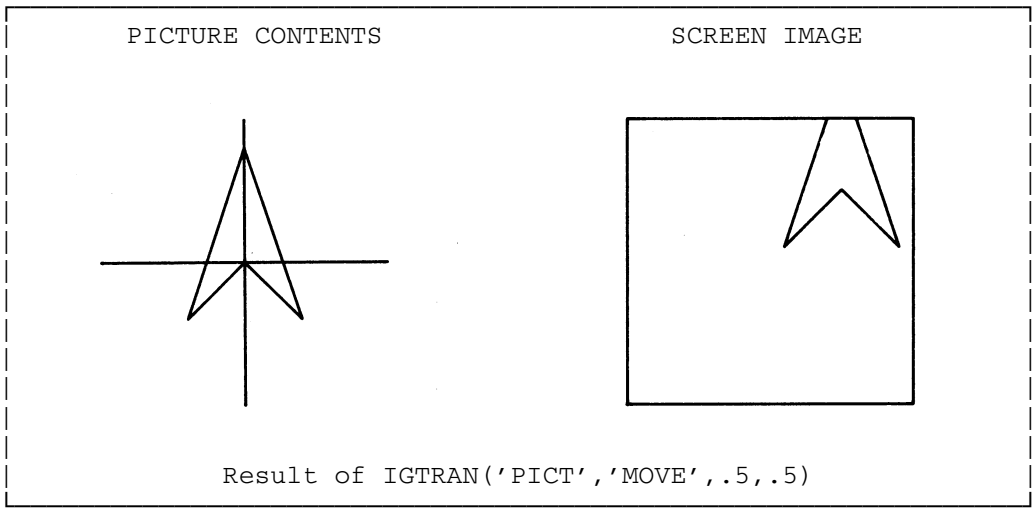
Basic Transformations in Two Dimensions

There are four basic transformation types: translation, rotation, scaling, and windowing.

Translation is specified by calling IGTRAN, giving 'MOVE' as the transformation type:

```
CALL IGTRAN(NAMPIC,'MOVE',X,Y)
```

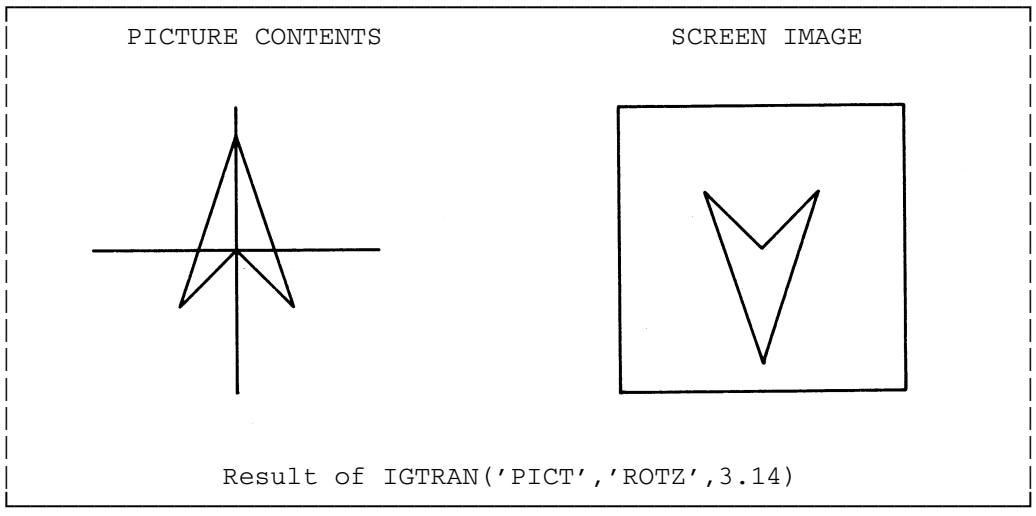
The picture NAMPIC is moved horizontally by a displacement X and vertically by a displacement Y. In other words, the picture coordinate (0,0) is mapped into the screen coordinate (X,Y).



Rotation is specified by calling IGTRAN, giving 'ROTZ' as the transformation type:

```
CALL IGTRAN(NAMPIC,'ROTZ',ANGLE)
```

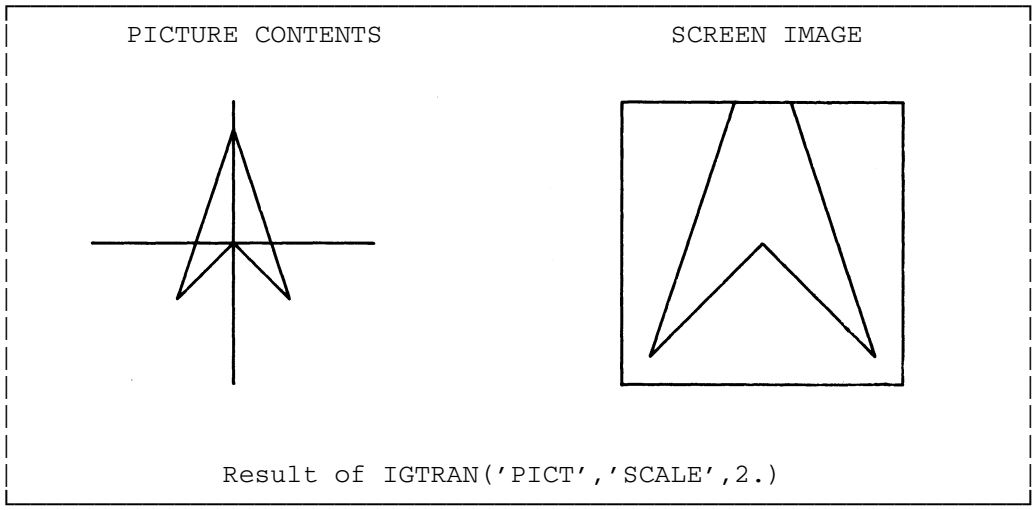
The picture NAMPIC is rotated clockwise through ANGLE radians, about its origin (0,0).



Scaling is specified by calling IGTRAN, giving 'SCALE' as the transformation type:

```
CALL IGTRAN(NAMPIC,'SCALE',FACTOR)
```

The picture NAMPIC is scaled by FACTOR. The origin of the picture remains fixed while the surrounding parts are expanded or contracted.



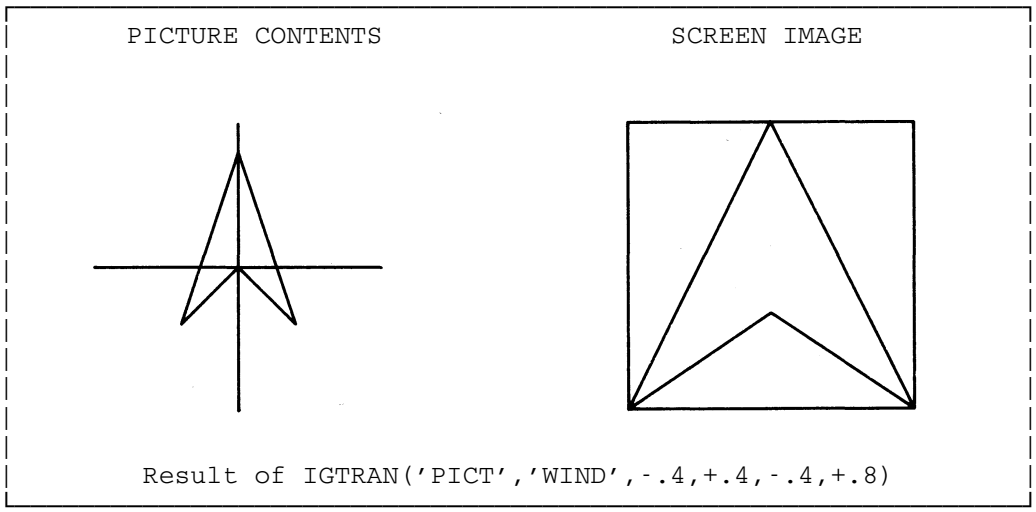
Windowing is a transformation which maps a rectangular subregion of the picture (a window) onto the screen. Windowing is specified by calling IGTRAN, giving 'WIND' as the transformation type:

```
CALL IGTRAN(NAMPIC,'WIND',XLEFT,XRIGHT,YBOTOM,YTOP)
```

In this case, the window is defined by XLEFT, XRIGHT, YBOTOM, and YTOP. The portion of the picture that resides in this window is mapped into the square region defined by -1,+1,-1,+1. This mapping consists of a translation followed by separate scalings of the X and Y coordinates.

Windowing allows the user program to describe a picture in any convenient coordinate system (using any floating-point values), then map the entire picture into the visible portion of the screen. Note that the mapping may distort the picture by scaling the X and Y coordinates separately. In some cases, this distortion may be useful, e.g., when plotting a curve in which the X extent is very different from the Y extent. In other cases, the user may wish to avoid this distortion by using the technique shown in Example Program 3 of Appendix C.

The user program may obtain the minimum and maximum X and Y coordinates by calling the IGINFO subroutine. See the IGINFO description in Appendix B.

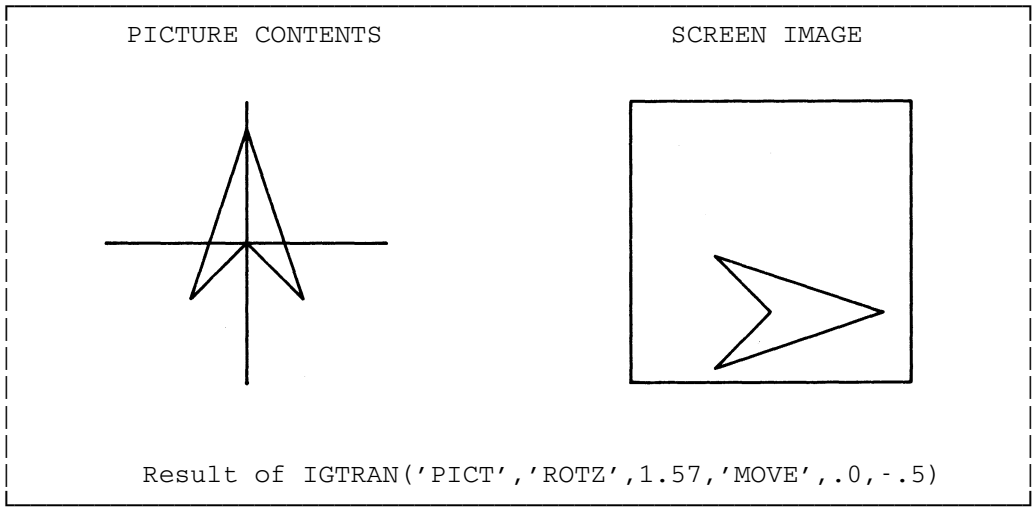


Composite Transformations

Several different basic transformations may be concatenated to form a composite transformation. This can be done by concatenating the IGTRAN transformation parameters. For example,

```
CALL IGTRAN(NAMPIC,'ROTZ',1.57,'MOVE',.0,-.5)
```

performs a rotation followed by a translation. The picture is first rotated 90° clockwise about (0,0) and then translated 0.5 downward.



The transformations are applied in the order in which they appear in the IGTRAN parameter list. Note that this order makes a difference. A translation followed by a rotation would produce a different result, because the rotation would still be done around (0,0).

A transformation is represented internally by a 4x4 matrix of floating-point values. A composite transformation is represented by a product of matrices of basic transformations.

Relative Transformations

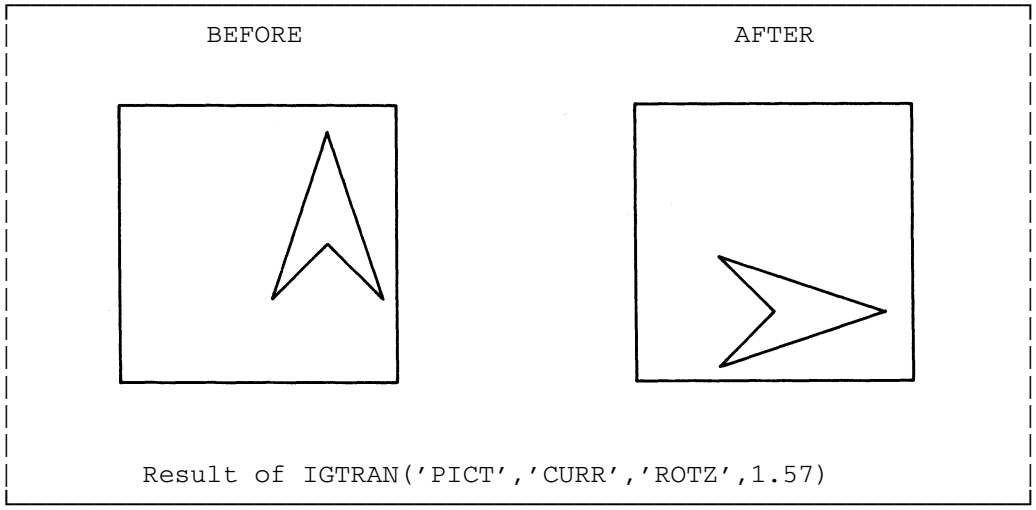
The current transformation of a picture is the transformation resulting from the most recent call to IGTRAN. The current transformation may by followed by additional transformations, thus forming a new composite transformation. (Internally, the matrix of the current transformation is multiplied by the matrices of the additional transformations.) For example,

```
CALL IGTRAN(NAMPIC,'CURR','MOVE',.5,.0)
```

specifies that the current transformation of NAMPIC is to be followed by a translation of 0.5 in the X direction. The keyword 'CURR' represents the current transformation. Again, the order of the parameters specifies the order in which the transformations are to be applied. For example,

```
CALL IGTRAN(NAMPIC,'CURR','ROTZ',ANGLE)
```

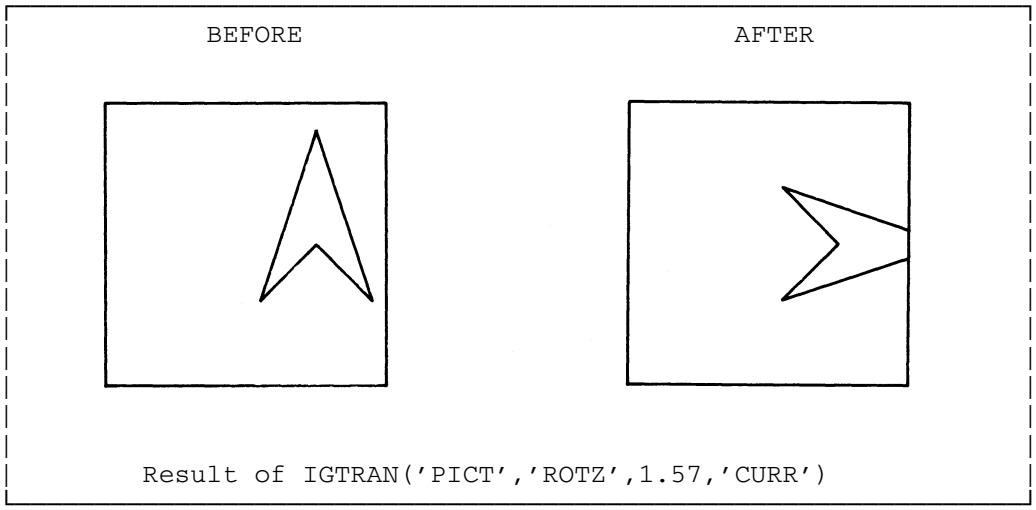
specifies that the current transformation of NAMPIC is to be followed by a rotation about the center of the screen.



On the other hand,

```
CALL IGTRAN(NAMPIC,'ROTZ',ALPHA,'CURR')
```

specifies that a rotation is to be applied to the picture before the current transformation is applied. Note that the order makes a difference.



Note that if the 'CURR' keyword is not included in a call to IGTRAN, then the new transformation will simply replace the current one. Thus, in the sequence of calls,

```
CALL IGTRAN('PICT','SCALE',.5)
CALL IGTRAN('PICT','MOVE',-.2,0.)
```

the 'MOVE' transformation will simply replace the 'SCALE' transformation. If these transformations are to be compounded, one of the following sequences must be used:

```
CALL IGTRAN('PICT','SCALE',.5,'MOVE',-.2,0.)
```

or

```
CALL IGTRAN('PICT','SCALE',.5)
CALL IGTRAN('PICT','CURR','MOVE',-.2,0.)
```

or

```
CALL IGTRAN('PICT','MOVE',-.2,0.)
CALL IGTRAN('PICT','SCALE',.5,'CURR')
```

Initial Specification of Transformations

A call to IGBGNS may include a list of IGTRAN parameters specifying a transformation for the new picture. For example,

```
CALL IGBGNS(NAMPIC,'MOVE',.75,.3,'SCALE',.1)
```

creates a picture, NAMPIC, and specifies the associated transformation.

PICTURE VIEWPORTS

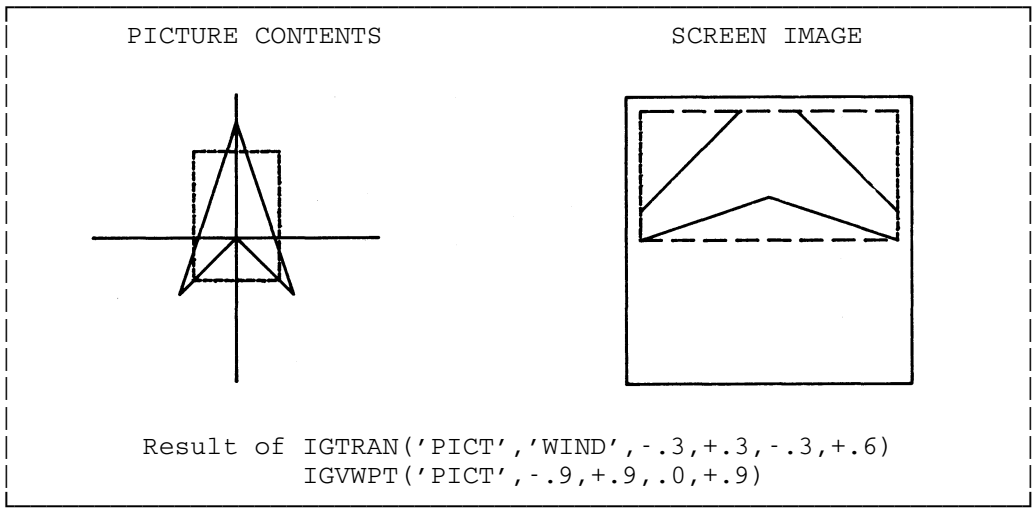
Viewporting is an operation that takes the portion of the picture that has been mapped into the square region delimited by -1,+1,-1,+1 and further maps it into a rectangular subregion of the screen (a viewport). The viewport becomes a "substitute screen". The portion of the picture that would have been displayed on the full screen is now displayed in the viewport.

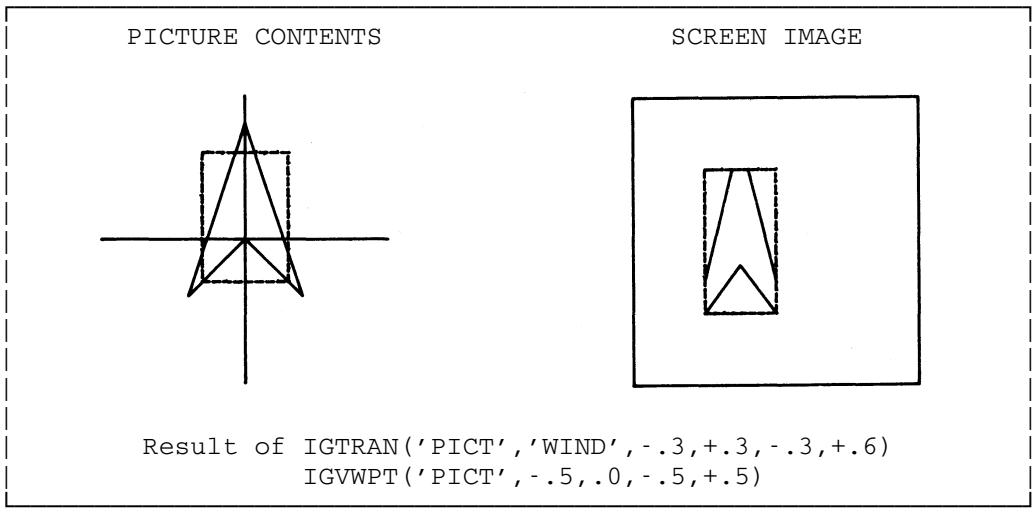
Viewporting is performed after any transformations specified by IGTRAN have been performed. By a combination of windowing and viewporting, any portion of a picture may be mapped into any portion of the screen. This is the traditional "window-to-viewport" operation.

Each picture has its own viewport, just as each picture has its own transformation. The viewport is stored with the picture but is independent of the contents of the picture and may be specified at any time before IGDRON is called. Separate pictures may be given separate viewports, so they can be displayed side-by-side on the screen (see Example Programs 2 and 3 in Appendix C). By default, each picture initially has the viewport -1,+1,-1,+1 (the whole screen). The viewport may be respecified by calling IGVWPT:

```
CALL IGVWPT(NAMPIC,XLEFT,XRIGHT,YBOTOM,YTOP)
```

This specifies that the viewport for NAMPIC is to be the rectangular subregion of the screen delimited by XLEFT, XRIGHT, YBOTOM, and YTOP. When IGDRON is called, NAMPIC will be transformed (according to transformations specified by IGTRAN) and then mapped into this viewport. The viewporting operation is similar to a translation by ((XR+XL)/2,(YT+YB)/2) and a scaling by ((XR-XL)/2,(YT-YB)/2). However, any lines extending outside of the viewport will be clipped off at the edges. In the following diagrams, the dotted lines delimit the window and viewport regions.





RESPECIFYING A PICTURE

The current contents of a picture may be replaced in the following way:

```
CALL IGBGNS(NAMPIC)
      .          ] calls to specify
      .          ] new lines and text
CALL IGENDS(NAMPIC)
```

If IGBGNS is called with the name of an existing picture, the picture is emptied of its current contents and is reactivated. The current

position is reset to the origin (0,0). New lines and text can then be added by calling picture-description subroutines. The picture transformation, viewport, and other attributes are not affected by this process. The picture is terminated by a call to IGENDS. Following this call, the picture that was active before the call to IGBGNS is reactivated.

ADDING TO A PICTURE

New lines or text may be added to an existing picture in the following way:

```
CALL IGCTNS(NAMPIC)
      .          ] calls to add
      .          ] new lines and text
CALL IGENDS(NAMPIC)
```

The call to IGCTNS reactivates the existing picture but does not alter its current contents. The current position remains the same as when the picture was last active. New lines and text can be added by calling picture-description subroutines. The picture is terminated by a call to IGENDS. Following this call, the picture that was active before the call to IGCTNS is reactivated.

DELETING A PICTURE

A picture may be deleted by calling IGDELS:

```
CALL IGDELS(NAMPIC)
```

The picture NAMPIC is deleted from the internal data structure. On the next call to IGDRON, NAMPIC will not be displayed.

PICTURE ATTRIBUTES

Associated with each picture are several picture attributes. These are independent from each other and independent from the picture contents (lines and text). The attributes are manipulated by calling various subroutines:

attribute	subroutine
transformation	IGTRAN
viewport	IGVWPT
pen number	IGATTS or IGATTB
default text scale	IGATTS or IGATTB
default character set or font	IGATTS or IGATTB
user word	IGUSER

The attributes may be manipulated at any time after the picture has been created. (The picture need not be active.) The attributes will be used by IGDRON in displaying the picture.

The transformation and viewport were described in the preceding section. This section describes the remaining attributes.

PEN NUMBER

The pen number may be any integer from 0 to 32767. This number is a code that represents various line qualities that can be produced by the output device (using various "pens"). For example, a given pen number might represent a specific hue or intensity level. The actual interpretation of pen numbers will depend on the type of output device (see Appendix I). The pen number for a picture is initially 1, but may be changed by calling IGATTS:

```
CALL IGATTS(NAMPIC,'PEN ',N)
```

NAMPIC is the name of the picture, 'PEN ' is the keyword giving the attribute (pen number), and N is the pen number (from 0 to 32767).

DEFAULT TEXT SCALE

The default text scale for a picture is initially the hardware text scale, but may be changed by calling IGATTS:

```
CALL IGATTS(NAMPIC,'TSCALE',SCALE)
```

NAMPIC is the name of the picture, 'TSCALE' is the keyword giving the attribute (default text scale), and SCALE is the new default text scale (in picture coordinates). This default text scale will be used for all text added by IGTXT or IGFMT, except text that is locally scaled (see the section "Picture-Description Subroutines").

DEFAULT CHARACTER SET OR FONT

The default character set or font for a picture is initially uppercase ASCII, but may be changed by calling IGATTS:

```
CALL IGATTS(NAMPIC,'FONT',CSET)
```

NAMPIC is the name of the picture, 'FONT' is the keyword giving the attribute (default character set or font), and CSET is a keyword giving the name of the new default character set or font:

'STANDARD'	uppercase ASCII
'7ASCII'	full 7-bit ASCII
'SANSERIF.1'	
'SANSERIF.2'	
'SANSERIF.CART'	
'ROMAN.2A'	
'ROMAN.2'	
'ROMAN.3'	
'ITALIC.2A'	
'ITALIC.2'	
'ITALIC.3'	
'SCRIPT.1'	
'SCRIPT.2'	
'GREEK.1'	
'GREEK.2A'	
'GREEK.2'	
'GREEK.CART'	
'GREEK'	
'GOTHIC.ENGLISH'	
'GOTHIC.FRAKTUR'	
'GOTHIC.ITALIAN'	
'CYRILLIC.2'	

These keywords must be spelled out in full. The default character set or font will be used for all text added by IGTXT, IGFMT, or IGSYM,

except text that is locally generated in a different character set or font (see the section "Picture-Description Subroutines").

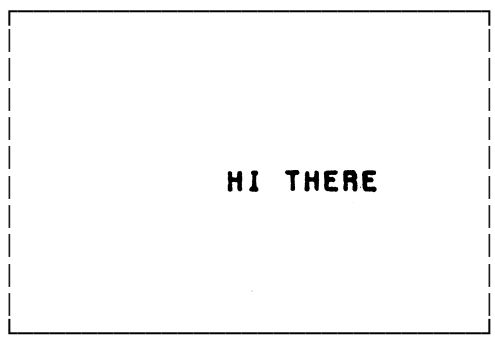
DEFAULT TEXT PLANE

The text plane is described by three floating-point numbers. Text strings for the subpicture are drawn along the same direction as the vector from the origin to the vector formed by the three floating-point numbers.

The following examples illustrate the use of IGATTS for describing the text plane:

```
      ...
     CALL IGATTS(name,'TPLANE',1.0,0.0,0.0)
     CALL IGTXT('HI THERE<E>')
      ...
```

produces

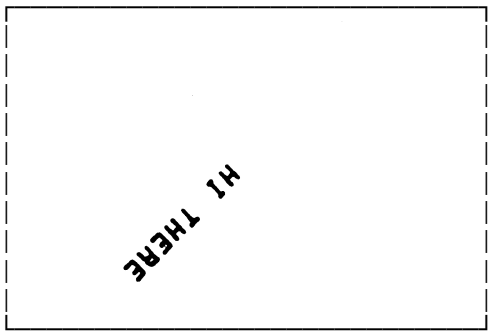



```
      ...
     CALL IGATTS(name,'TPLANE',0.0,1.0,0.0)
     CALL IGTXT('HI THERE<E>')
      ...
```

produces


```
 ...
CALL IGATTS(name,'TPLANE',-1.0,-1.0,0.0)
CALL IGTXT('HI THERE<E>')
 ...
```

produces



USER WORD

The user word for a picture is a single word (four bytes) that is set aside for use by the user. Typically, this word might be used as a pointer to a block of auxiliary information describing the picture. The contents of the user word may be stored or fetched by calling IGUSER. The keyword 'PUT ' specifies that a value is to be stored in the user word. The call

```
CALL IGUSER(NAMPIC,'PUT ',X)
```

stores the value of the variable X in the user word for NAMPIC. On the other hand, the keyword 'GET ' specifies that a value is to be retrieved from the user word. The call

```
CALL IGUSER(NAMPIC,'GET ',X)
```

fetches the current value of the user word and assigns it to the variable X.

SPECIFYING SUBPICTURE ATTRIBUTES

There are two ways to specify attributes for subpictures, either by using the routine IGATTS or the routine IGATTB. The attributes set using IGATTS affect the entire picture and are inherited by lower-level subpictures. If no attributes are explicitly specified by IGATTS, they will be inherited from the next higher-level subpicture. Attributes can be set by IGATTS anytime after the subpicture has been created. The routine IGATTB will allow attributes to be dynamically changed during the definition of the subpicture. The change will affect additions to the subpicture after the call to IGATTB. This is a way for example to use two different colors in one subpicture. Attributes set using the IGATTB routine will not be inherited by lower-level subpictures.

COPYING ATTRIBUTES FROM OTHER PICTURES

The subroutine IGLIKE can be used to copy attributes from other pictures. For example,

```
CALL IGLIKE(NAMPIC,'TRAN','PIC1','PEN ','PIC2')
```

copies attributes from two other pictures to NAMPIC. The attribute 'TRAN' (transformation) is copied from the picture 'PIC1' and the attribute 'PEN ' (pen number) is copied from the picture 'PIC2'. In general, each attribute keyword must be followed by the name of the picture from which it is to be copied. The legal attribute keywords are:

'TRAN' or 'XFRM'	transformation
'VWPT' or 'VIEW'	viewport
'PEN '	pen number
'TSCALE'	default text scale
'FONT'	default character set or font
'EVERYTHING'	transformation, viewport, pen number, default text scale, and default character set or font

As with most IG keywords, more than four characters may be given (e.g., 'TRANSFORMATION') but only the first four will be used. The parameters in calls to IGLIKE are processed from left to right. For example,

```
CALL IGLIKE(NAMPIC,'EVER','PIC1','PEN ','PIC2')
```

specifies that NAMPIC is to get all its attributes from 'PIC1', and then get its pen number from 'PIC2'.

SUBPICTURES

A picture may have several subpictures. Each subpicture may be transformed within the coordinate space of the higher-level picture. Then, the higher-level picture and its subpictures may be transformed as a unit.

As an example, consider a picture that represents an airplane (see Figure 7). The propeller of the airplane may be represented as a subpicture. This subpicture may be rotated within the coordinate space of the higher-level picture. Then, the higher-level picture (the whole airplane) may be moved as a unit.

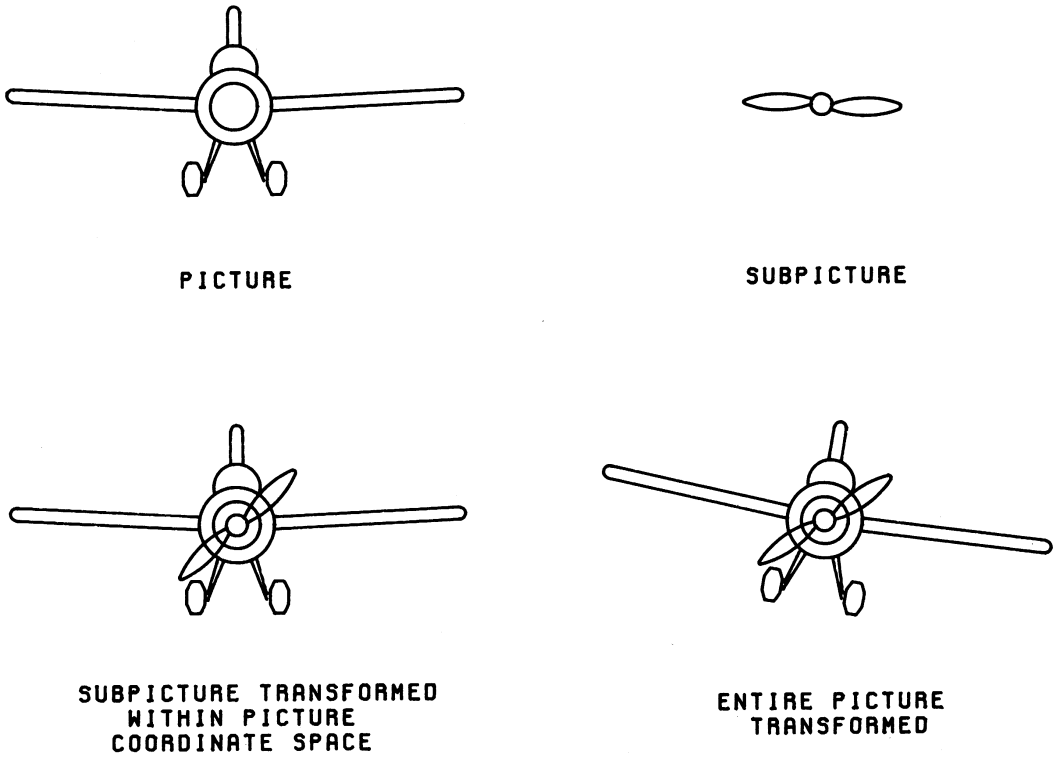
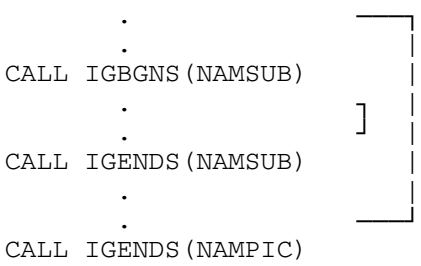


Figure 7: Subpictures

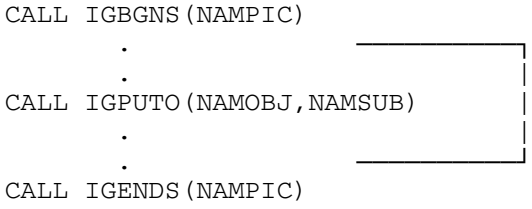
CREATING SUBPICTURES

There are two ways to create subpictures. The first way is to nest the IGBGNS...IGENDS calls creating the subpicture within the IGBGNS... IGENDS calls generating the higher-level picture:



The call IGBGNS(NAMSUB) creates a subpicture of NAMPIC. Following this call, NAMSUB becomes the active subpicture. Its current position is set to the origin (0,0). Lines and text may be added by calling picture-description subroutines. The call IGENDS(NAMSUB) terminates the subpicture. Following this call, NAMPIC again becomes the active picture.

The second way to create subpictures is to call the subroutine IGPUTO while the higher-level picture is active:



This creates a subpicture NAMSUB that is an *instance* of the *object* NAMOBJ. This procedure is covered in more detail in the following section, "Objects".

Each subpicture is also a picture in its own right. The term "subpicture" is used only to emphasize its relationship to a higher-level picture.

The highest-level picture in the IG data structure is the *main picture*, which can be referenced by the special name '*MP*'. Following a call to IGINIT, '*MP*' becomes the active picture. Subpictures of '*MP*' are created by nonnested calls to IGBGNS.

The currently active picture can be referenced by the special name '*AP*'.

A subpicture may itself have subpictures. The IG data structure is a treelike structure that can have any degree of complexity (see Appen-

dix G). One or two levels of subpictures are sufficient for most applications.

TRANSFORMING SUBPICTURES

Each subpicture has an associated transformation (specified by calls to IGTRAN), which maps the subpicture into the coordinate space of the next-higher-level picture. This picture also has an associated transformation, which maps it into the coordinate space of the next-higher-level picture, and so on. Thus, the transformation that maps a subpicture onto the screen consists of its own transformation (as specified by calls to IGTRAN) followed by the transformations of successively higher-level pictures, up through and including the transformation of the main picture '*MP*'.

SUBPICTURE VIEWPORTS

Each subpicture has an associated viewport. This viewport is always defined in screen coordinates (not in the coordinates of the next-higher-level picture). The effective viewport that is actually used to display the subpicture consists of the viewport specified by a call to IGVWPT intersected with the effective viewport of the next-higher-level picture (see Figure 8).

In most applications, only subpictures of the main picture will have viewports specified by IGVWPT. Lower-level subpictures will have default viewports -1,+1,-1,+1. By the intersection rule, such a lower-level subpicture will have the same effective viewport as the next-higher-level picture.

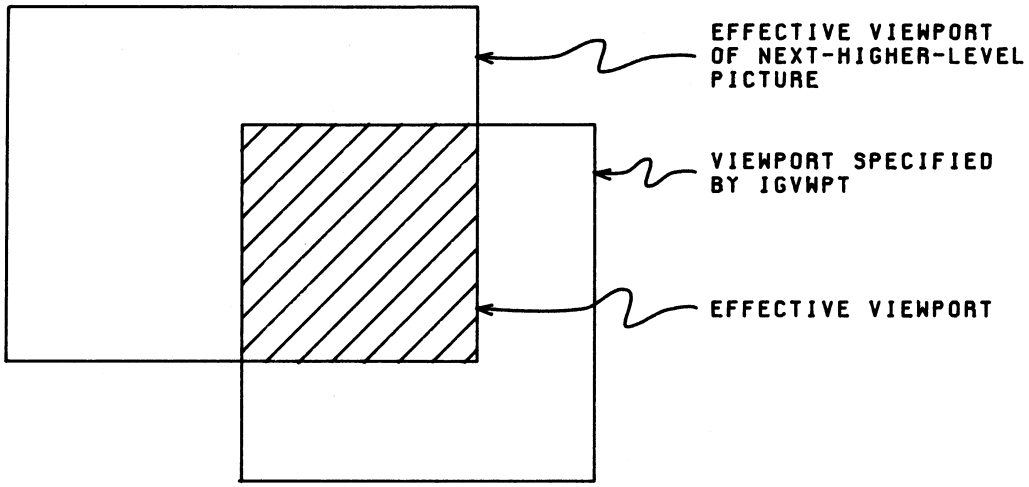


Figure 8: Subpicture Viewport

SUBPICTURE ATTRIBUTES

In addition to a transformation and a viewport, a subpicture has several other attributes. These include the pen number, text plane, default text scale, and default character set or font. If these attributes are not explicitly specified using the routine IGATTS, they will be inherited from the next higher-level subpicture.

DELETING A SUBPICTURE

A subpicture may be emptied by calling IGBGNS or deleted by calling IGDELS. In either case, all of its subpictures will also be deleted.

OBJECTS

An object is a data unit that contains lines and text. The contents of an object are specified in the same way as the contents of a picture, i.e., by calling picture-description subroutines. However, the contents of an object are reproducible; by calling the subroutine IGPUTO, the user program may create a number of subpictures that are instances (copies) of the object. Since these have transformations, each one may be placed at a different location and at a different orientation (see Figure 9). If the contents of an object are altered, each instance is altered accordingly. Operations on objects are illustrated in Example Program 5 of Appendix C.

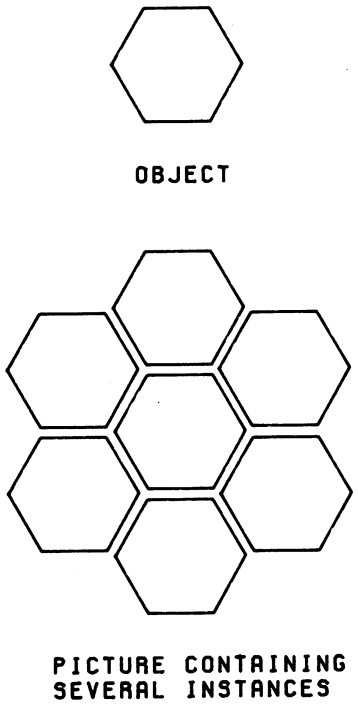


Figure 9: Instances of an Object

CREATING OBJECTS

An object is created by a call to IGBGNO:

```
CALL IGBGNO(NAMOBJ)
      .          ] calls to picture-
      .          ] description subroutines
CALL IGENDO(NAMOBJ)
```

NAMOBJ must contain a four-character object name which must begin with a letter. The object name is usually expressed in the form of a literal string. If the string has more than four characters, only the first four will be used. Following the call to IGBGNO, NAMOBJ becomes the active object. Lines and text may be added by making calls to picture-description subroutines. The object is terminated by a call to IGENDO, giving the same parameter as the call to IGBGNO. This parameter serves as an error check against unmatched IGBGNO...IGENDO pairs.

The user program may also create objects dynamically, i.e., according to conditions that exist at execution time. Instead of supplying object names, the user program may call IGBGNO as a function, giving 0 as the parameter:

```
INTNAM = IGBGNO(0)
      .          ] calls to picture-
      .          ] description subroutines
CALL IGENDO(INTNAM)
```

IG will generate a unique, internal name for each new object and return it as the function value of IGBGNO. This name will be some large, positive integer.

Each object has its own current position. Following a call to IGBGNO, the current position is set to the origin (0,0).

CREATING INSTANCES OF AN OBJECT

Unlike a subpicture, an object has no transformation, viewport, or other attributes. Furthermore, an object is not displayed when IGDRON is called. An object serves only as a pattern from which a number of instances can be reproduced. Each instance is a subpicture and thus has a transformation, viewport, and other attributes, and is displayed when IGDRON is called.

An instance of an object is created by calling IGPUTO:

```
CALL IGPUTO(NAMOBJ,NAMSUB)
```

NAMSUB must contain a four-character subpicture name which must begin with a letter. The subpicture name is usually expressed in the form of a literal string. If the string has more than four characters, only the first four will be used. NAMSUB is created as a subpicture of the active picture and as an instance of the object NAMOBJ.

The user program may also create instances dynamically. Instead of supplying subpicture names, the user program may call IGPUTO as a function, either giving the second parameter as 0 or omitting it:

```
INTNAM = IGPUTO(NAMOBJ,0)
```

or

```
INTNAM = IGPUTO(NAMOBJ)
```

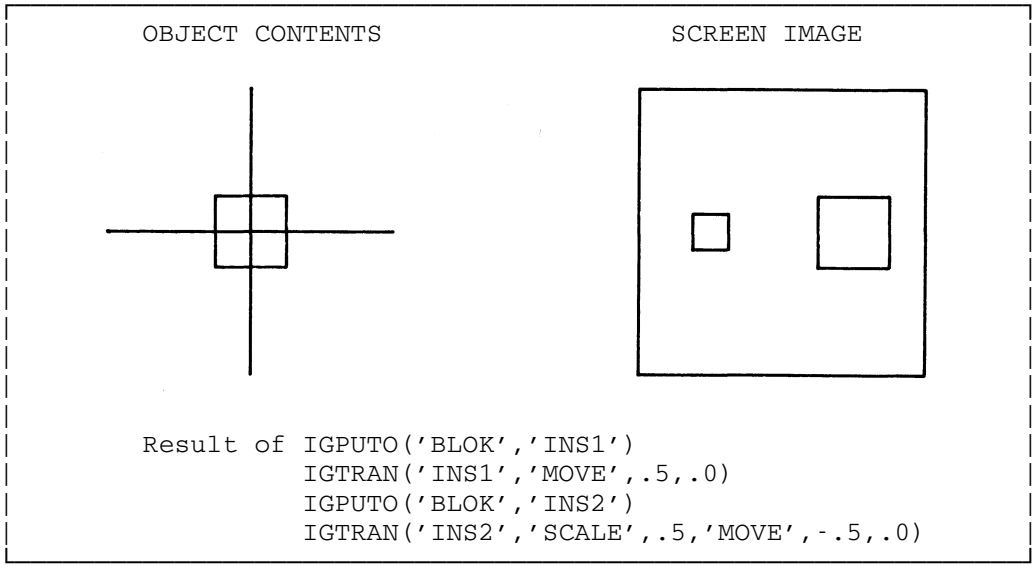
IG will generate a unique, internal name for each new subpicture and return it as the function value of IGPUTO.

TRANSFORMING INSTANCES OF AN OBJECT

An instance of an object is a subpicture and thus can be transformed within the coordinate space of the next-higher-level picture. For example, the calls

```
CALL IGPUTO('BLOK','INS1')
CALL IGTRAN('INS1','MOVE',.5,.0)
CALL IGPUTO('BLOK','INS2')
CALL IGTRAN('INS2','SCALE',.5,'MOVE',-.5,.0)
```

create two instances of the object 'BLOK', one at (.5,.0) and the other, half as big, at (-.5,.0).



Since instances are commonly transformed, a call to IGPUTO may include a list of IGTRAN parameters specifying a transformation. For example, the calls

```
CALL IGPUTO('BLOK','INS1','MOVE',.5,.0)
CALL IGPUTO('BLOK','INS2','SCALE',.5,'MOVE',-.5,.0)
```

have the same effect as the calls in the preceding example. In general, the second through last parameters to IGPUTO have the same format as the first through last parameters to IGBGNS.

DELETING INSTANCES OF AN OBJECT

An instance of an object (but not the object itself) may be deleted by calling IGDELS on the subpicture created by IGPUTO. For example, if the instance was created by the call

```
CALL IGPUTO('OBJT','INST')
```

then it may be deleted by the call:

```
CALL IGDELS('INST')
```

The subpicture 'INST' is deleted from the internal data structure. On the next call to IGDRON, 'INST' will not be displayed.

DELETING AN OBJECT

An object itself may be deleted by calling IGDELO:

```
CALL IGDELO(NAMOBJ)
```

The object NAMOBJ (created by IGBGNO) is deleted from the internal data structure. Furthermore, all instances (created by IGPUTO) of the object are also deleted.

RESPECIFYING AN OBJECT

The current contents of an object may be respecified in the following way:

```
CALL IGBGNO(NAMOBJ)
       .            ]  calls to specify
       .            ]  new lines and text
CALL IGENDO(NAMOBJ)
```

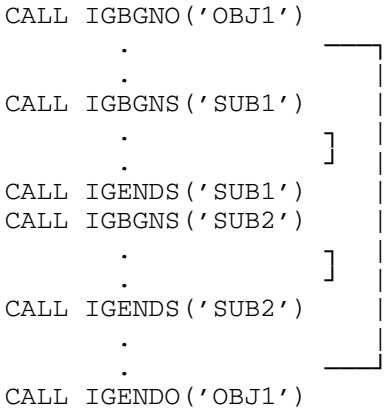
If IGBGNO is called with the name of an existing object, the object is emptied of its current contents and is reactivated. The current position is reset to the origin (0,0). New lines and text can then be added by calling picture-description subroutines. When an object is respecified in this way, all instances of the object are respecified accordingly. The object is terminated by a call to IGENDO. Following this call, the picture or object that was active before the call to IGBGNO is reactivated.

MODIFYING AN OBJECT

The contents of an object are modified whenever the contents of any instance are modified. Internally, an instance does not have its own contents, but it does have a pointer to the contents of the object. Thus, when any instance is modified (by calling IGBGNS or IGCTNS), the object is modified and all instances are modified accordingly.

SUBPICTURES WITHIN AN OBJECT

An object may have several subpictures. These may be created by nesting IGBGNS...IGENDS calls within the IGBGNO...IGENDO calls:



In this case, the object 'OBJ1' contains two subpictures, 'SUB1' and 'SUB2'. These subpictures have their own transformations, and may be manipulated within the coordinate space of the object. When a subpicture is manipulated, the object is changed and all instances of the object are changed accordingly.

GRAPHIC INPUT

Most graphics terminals provide a tracking cross, cursor, pair of crosshairs, or some such locator that can be moved to any position on the screen (see Figure 10). The specific mechanism for moving the locator depends on the terminal type (see Appendix I). When a keyboard key is pressed, the locator position is returned to the executing program. This kind of input is known as graphic input. It is often used for interactive, visually directed operations on the screen image.

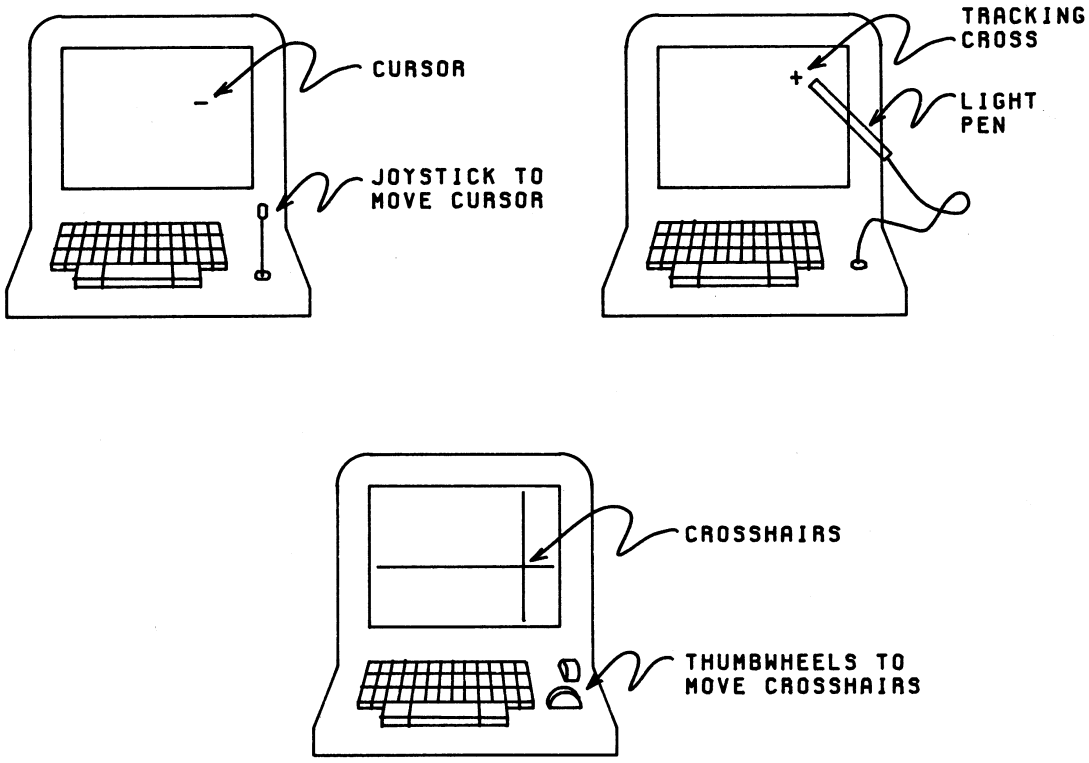


Figure 10: Typical Graphic Input Mechanisms

COORDINATE INPUT

The user program may read the locator coordinates by calling the subroutine IGXYIN:

```
CALL IGXYIN(XIN,YIN)
```

This subroutine implicitly calls IGDRON('TERMINAL') to update the screen image. It then pauses and waits for the user to position the locator. When any keyboard key is pressed, the locator coordinates are returned in XIN and YIN. The locator coordinates are expressed in the coordinate system of the active picture, '*AP*'. Thus, for example, if the locator coincides with the origin of the active picture as displayed on the screen (no matter what transformation or viewport has been applied) then the locator coordinates will be returned as (0,0). If neither a transformation nor a viewport has been applied, then the picture coordinates are the same as the screen coordinates. Note that if the main picture '*MP*' is active, then the locator coordinates are expressed in the coordinate space of the main picture.

In one typical application, the coordinates returned by IGXYIN are passed to IGDA or IGMA, to perform a draw or move from the current position to the locator position. In another typical application, the coordinates are passed as 'MOVE' transformation parameters to IGPUTO, to create an instance, at the locator position, of an object.

If IGXYIN is called as a function, it will return a code for the keyboard key that was pressed to return the locator position.

```
ICODE = IGXYIN(XIN,YIN)
```

The code value can be used to indicate the interpretation of the coordinate values. For example, ICODE=0 could indicate that IGMA(XIN, YIN) is to be called, and ICODE=1 could indicate that IGDA(XIN,YIN) is to be called. The correspondence between ICODE values and keys is given in Table 1. Note that a given ICODE value is obtained from different keys on different terminals. Those keys that are most accessible (the program function keys on the IBM 3270, the Fn buttons on a Computek 400, or the top row of keys on a Tektronix 4010) translate into the same codes--the integers 1 through 6. Any keys which do not appear in Table 1 translate into 0.

ICODE	char	3270 key	CK400 key	ICODE	char	ICODE	char	ICODE	char
0	0			16	@	32	Pp	48	(blank)
1	1	PF1	F1	17	Aa	33	Qq	49	!
2	2	PF2	F2	18	Bb	34	Rr	50	"
3	3	PF3	F3	19	Cc	35	Ss	51	#
4	4	PF4	F4	20	Dd	36	Tt	52	$
5	5	PF5	F5	21	Ee	37	Uu	53	%
6	6	PF6	F6	22	Ff	38	Vv	54	&
7	7	PF7		23	Gg	39	Ww	55	’
8	8	PF8		24	Hh	40	Xx	56	(
9	9	PF9		25	Ii	41	Yy	57	)
10	:			26	Jj	42	Zz	58	*
11	;			27	Kk	43	[	59	+
12	<			28	Ll	44	\	60	,
13	=			29	Mm	45	]	61	-
14	>			30	Nn	46	(caret)	62	.
15	?			31	Oo	47	_	63	/

Table 1: Return Codes from IGXYIN

PICTURE PICKING VIA USER INPUT

The user may pick one of several pictures displayed on the screen by using the terminal’s pick locator. Depending on the terminal type, this may be a light pen, cursor, pair of crosshairs, or some other mechanism (see Appendix I). The pick locator may or may not be the same mechanism as the locator that is used for coordinate input. Picking is initiated by a call to IGPIKS:

INDEX = IGPIKS(NAME1,NAME2,...,NAMEn)

NAME1,...,NAMEn is a list of pictures, supplied by the user program, from which one is to be picked. The subroutine implicitly calls IGDRON(’TERMINAL’) to update the screen image, then pauses and waits for the user to pick one of the pictures by hitting it with the pick locator (i.e., touching it or nearly touching it). The subroutine then returns an index value 1,...,n (in INDEX) to indicate which of the pictures NAME1,...,NAMEn was picked. If none of the pictures NAME1,...,NAMEn was picked, the subroutine returns the value 0.

In one typical application, the user program modifies or deletes the picture that was picked. In another typical application, the user program displays a menu of items on the screen of the terminal. These items represent various operations that can be performed by the program (see Figure 11). By picking an item from the menu, the user can select an operation. See Example Program 4 in Appendix C.

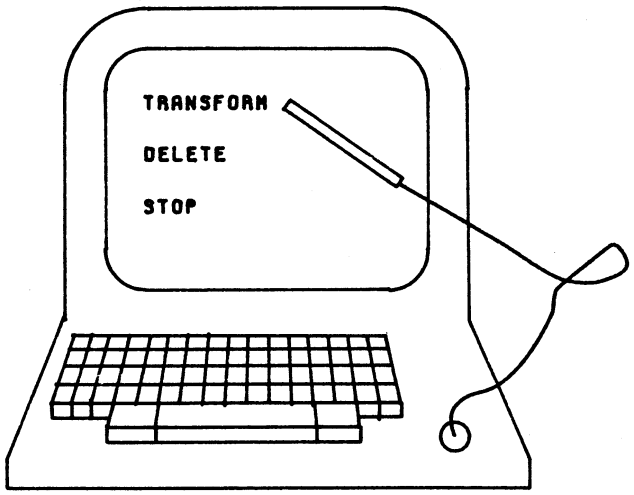


Figure 11: Picking an Item from a Menu

PICTURE PICKING VIA PROGRAM INPUT

The user program may pick pictures without user intervention, in a manner that simulates the use of a pick locator. This is done by calling the subroutine IGPIKC:

INDEX = IGPIKC(CPIC,XVAL,YVAL,NAME1,NAME2,...,NAME*n*)

The user program supplies a picture name CPIC and a pair of coordinates XVAL and YVAL in the coordinate space of this picture. These coordinates are mapped into screen coordinates by applying the transformation and viewport associated with CPIC. The resulting screen coordinates represent a "locator" position. Note that the IGPIKC subroutine starts with a pair of picture coordinates and generates a locator position. (This is the reverse of the procedure performed by the IGXYIN subroutine, which starts with a locator position and generates a pair of picture coordinates.) Once IGPIKC has generated a locator position, it proceeds in a manner similar to IGPIKS. NAME1,...,NAME*n* is a list of picture names, supplied by the user program, from which one is to be picked. The subroutine returns an index value 1,...,*n* (in INDEX) to indicate which of the pictures was picked. If none of the pictures was picked, the subroutine returns the value 0.

MORE ABOUT PICTURE PICKING

Each subpicture in the IG data structure has a nesting level. Subpictures of the main picture '*MP*' have the nesting level 0, subpictures of these subpictures have the nesting level 1, and so on. Successively higher nesting-level numbers represent successively lower nesting levels.

On a call to IGPIKS or IGPIKC, a picture will be picked if any of its subpictures (on any nesting level) is hit by the pick locator. The subroutine IGPIKN may be called to obtain additional information about the hit:

```
NAMSUB = IGPIKN(LEVEL,XHIT,YHIT)
```

LEVEL is the nesting level, supplied by the user program, for which information is desired. If the hit occurred on a lower (numerically higher) nesting level, it is also considered to have occurred on LEVEL. The information returned includes NAMSUB, the name of the subpicture (at LEVEL) on which the hit occurred, and XHIT and YHIT, the coordinates of the hit in the coordinate space of NAMSUB. If no hit occurred on LEVEL (or any lower nesting level), the returned function value will be 0.

Note that repeated calls to IGPIKN at successively lower (numerically higher) nesting levels will eventually determine the lowest-level subpicture that was hit.

Note also that IGPIKN only returns information about the most recent call to either IGPIKS or IGPIKC.

THREE-DIMENSIONAL PICTURES

A three-dimensional picture (or object) is generated by passing Z coordinates as well as X and Y coordinates to the picture-description subroutines. The resulting lines are "drawn" in the three-dimensional coordinate space of the picture. When IGDRON is called to display the picture, the associated three-dimensional transformation is applied. Then the picture is projected orthographically onto the XY plane (which corresponds to the screen). This projection is performed by merely ignoring the Z coordinates (see Figure 12). Finally, the portion of the XY plane delimited by -1,+1,-1,+1 is mapped into the viewport associated with the picture.

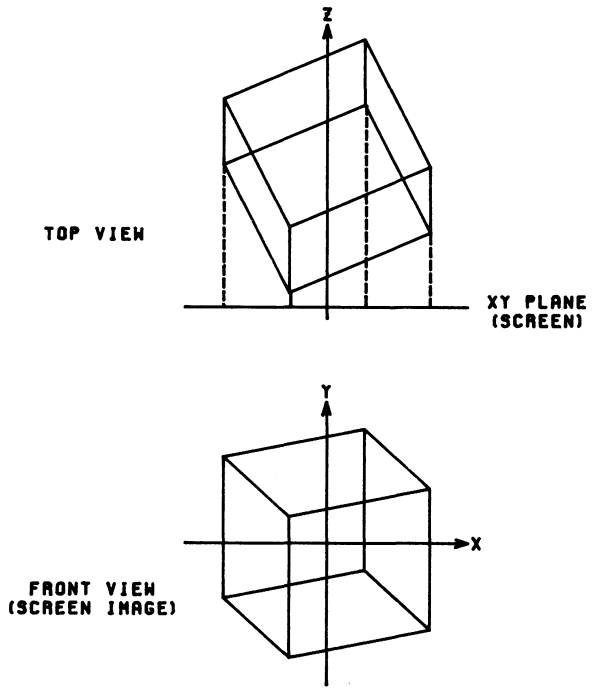


Figure 12: Orthographic Projection

The XYZ picture coordinates are left-handed, with the XY plane corresponding to the plane of the screen and the Z axis pointing into the screen. If Z coordinates are not passed to the picture-description subroutines, the resulting picture will be confined to the XY plane and will be two-dimensional. Thus, a two-dimensional picture can be regarded as a special case of a three-dimensional picture. Note, however, that even if Z coordinates are passed, they will not be visible unless the picture is transformed in some appropriate way (e.g., rotated about the X axis). Otherwise, the Z coordinates will merely collapse back into the XY plane under the orthographic projection.

The orthographic projection can be changed to a perspective projection by making the appropriate calls to IGTRAN (see below). Under a perspective projection, the Z coordinates will generally be visible.

LINES IN THREE DIMENSIONS

When a picture or object is created, its current position is set to (0,0,0). Lines may be added to the picture or object by calling the following subroutines:

CALL IGMA(X,Y,Z)

Move (draw an invisible line) from the current position to the absolute position (X,Y,Z).

CALL IGDA(X,Y,Z)

Draw a visible line from the current position to the absolute position (X,Y,Z).

CALL IGMR(DX,DY,DZ)

Move with a displacement (DX,DY,DZ) relative to the current position.

CALL IGDR(DX,DY,DZ)

Draw with a displacement (DX,DY,DZ) relative to the current position.

CALL IGVEC(NDIM,INTVEC,MOVTYP,NWORDS,XVEC,YVEC,ZVEC)

Add NDIM lines, obtaining coordinates from XVEC, YVEC, and ZVEC (see the section "Picture-Description Subroutines").

If the Z coordinates are omitted, then the moves and draws are done in the X and Y directions only. For example, if the current position is (X1,Y1,Z), then

```
CALL IGDA(X2,Y2)
```

will draw a line from the current position to the absolute position (X2,Y2,Z). If the Z coordinates are always omitted during the generation of a picture, then the Z coordinates will always be zero and the picture will be two-dimensional.

TRANSFORMATIONS IN THREE DIMENSIONS

All of the two-dimensional transformations that were described in the section "Picture Manipulations" may also be applied to three-dimensional pictures. If this is done, the X and Y coordinates will be transformed but the Z coordinates will not be affected.

There are several additional basic transformations that operate in three dimensions. These transformations may be concatenated in the manner described in the section "Picture Manipulations".

Translation in Three Dimensions

Translation is specified by calling IGTRAN, giving 'MOV3' as the transformation type:

```
CALL IGTRAN(NAMPIC,'MOV3',X,Y,Z)
```

The picture NAMPIC is moved by a displacement (X,Y,Z). In particular, the coordinate (0,0,0) is mapped into the coordinate (X,Y,Z).

Rotation in Three Dimensions

Rotation is specified by calling IGTRAN, giving 'ROTX', 'ROTY', or 'ROTZ' as the transformation type:

```
CALL IGTRAN(NAMPIC,'ROTX',ANGLE)

CALL IGTRAN(NAMPIC,'ROTY',ANGLE)

CALL IGTRAN(NAMPIC,'ROTZ',ANGLE)
```

The picture NAMPIC is rotated about the X, Y, or Z axis through ANGLE radians. The direction of rotation is determined by the right-hand rule: when the thumb of the right hand is pointed in the positive direction of the axis, the fingers will curl in the positive direction of rotation.

Perspective Projection

Perspective projection differs from orthographic projection in that the projection lines converge on a single viewpoint on the -Z axis, which represents the position of the viewer's eye (see Figure 13).

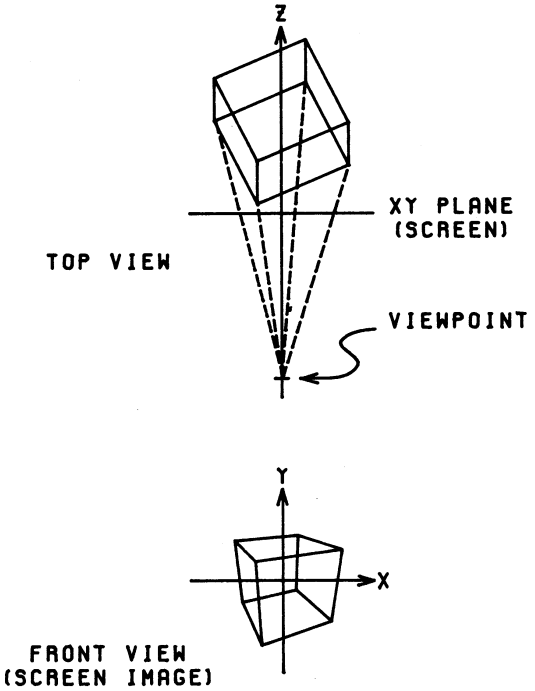


Figure 13: Perspective Projection

Perspective projection may be specified for a picture by calling IGTRAN, giving 'PROJ' as the transformation type:

```
CALL IGTRAN(NAMPIC,'PROJ',1./F,1./D)
```

F is the maximum visible Z coordinate. This means that any lines extending beyond the plane Z=F will be clipped. Typically, F will be infinite and 1./F will be 0. D is the distance from the viewpoint to the screen. This distance determines the severity of the perspective effect. Typically, D will be about 2.0 and 1./D will be about 0.5. As D approaches infinity and 1./D approaches 0, the perspective projection degenerates into the orthographic projection.

Before the perspective projection is performed, all lines extending in front of the screen (into the half space Z<0) are clipped and all lines extending beyond the plane Z=F are clipped. Furthermore, all lines extending outside of the pyramid with vertex at the viewpoint and edges passing through the screen corners (see Figure 14) are clipped. For more details, see I.E. Sutherland and G.W. Hodgman, "Reentrant Polygon Clipping", in Communications of the Association for Computing Machinery 17, 1 (January 1974), 32-42.

Normally, perspective projection should be performed after any other three-dimensional transformations. In other words, the 'PROJ' transformation should appear last in a concatenated series of transformations.

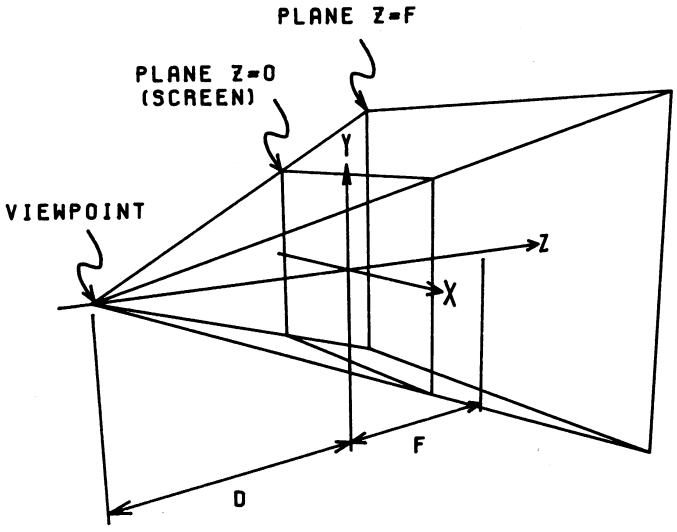
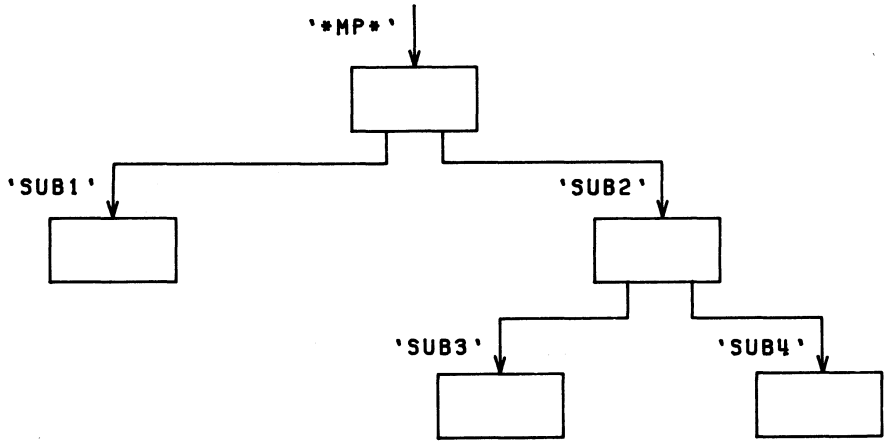


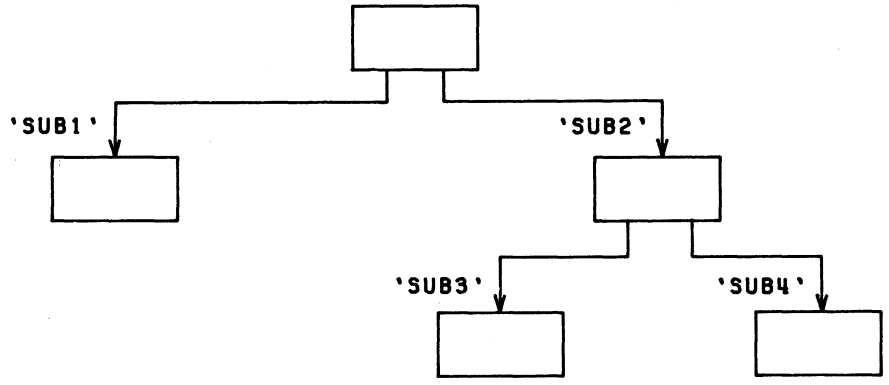
Figure 14: Clipping Volume for Perspective Projection

SAVING THE CURRENT DATA STRUCTURE AS AN OBJECT

The current data structure consists of the main picture '*MP*' and all subpictures (see Appendix G):



The current data structure also includes a name, transformation, viewport, and other attributes for each subpicture. If the main picture is stripped of its transformation, viewport, and other attributes,



the result is an object that represents most of the current data structure. (The current data structure does not include any objects which were created by IGBGNO but not attached to the data structure by IGPUTO.) This object can be saved in a file or other storage device by calling IGDRON with the keyword 'SAVE':

```
CALL IGDRON('SAVE')
```

Several objects, representing successive states of the data structure, may be saved in this file by making successive calls to IGDRON('SAVE'). A name for the next object to be saved with calls to IGDRON may be specified by calling IGCTRL with the keywords 'SAVE' and 'NAME' and a parameter giving the name:

```
CALL IGCTRL('SAVE','NAME','DS01')
```

The next object to be saved will be assigned the name 'DS01'. The first time a name is not specified, the default name '0000' (four zeros) will be assigned. Subsequently, every time a name is not specified, the default name will be incremented by 1 and assigned. Instead of saving the entire data structure, the routine IGSAVE can be used to save an object or subpicture:

```
CALL IGSAVE('SUB2')
```

The object is dumped, by default, on the logical I/O unit SPUNCH. A different destination for future objects may be specified by calling the subroutine IGCTRL with the keywords 'SAVE' and 'OUTPUT' and a parameter giving an FDname:

```
CALL IGCTRL('SAVE','OUTPUT','IGFILE ')
```

The FDname must be followed by a trailing blank. In this example, the FDname is a file name.

RELOADING OBJECTS

Objects that have been saved by calling IGDRON('SAVE') or IGSAVE may be reloaded by calling the subroutine IGLOAD, specifying the source FDname as the first parameter:

```
CALL IGLOAD('IGFILE ')
```

The FDname must be followed by a trailing blank. IGLOAD reads from this FDname and reloads all the objects it finds there, stopping when it reaches an end-of-file.

An object reloaded by IGLOAD will replace an existing object (if it has the same name as an existing object) or become a new object. In the latter case, the new object may be attached to the data structure by calling IGPUTO.

A reloaded object generally will have a hierarchy of subpictures. Since there is a possibility that the names of these subpictures will conflict with existing names, the names of these subpictures will normally be changed to arbitrary names. However, the names of reloaded

72.2 Saving the Current Data Structure as an Object

subpictures may be preserved by specifying the 'PRESERVE' keyword in the call to IGLOAD:

```
CALL IGLOAD('IGFILE ','PRESERVE')
```

Warning: If a reloaded subpicture has the same name as an existing subpicture, the contents of the existing subpicture will be replaced by the contents of the reloaded subpicture. Furthermore, the reloaded subpicture will be deleted from the reloaded object. Thus, the data structure may be altered significantly.

If several objects are to be reloaded, the names of these objects can be obtained by adding two parameters to the call to IGLOAD:

```
CALL IGLOAD('IGFILE ',N,LIST)
```

The number of objects is returned in N while the names of the objects are returned in the array LIST. (This array must be of sufficient dimension to hold all of the names.) The 'PRESERVE' keyword may also be specified:

```
CALL IGLOAD('IGFILE ',N,LIST,'PRESERVE')
```

This keyword must appear last in the parameter list.

DEVICE-DEPENDENT OPERATIONS

This section describes control operations that apply specifically to certain types of terminals or other devices. A control operation may be performed by making a call to the subroutine IGCTRL:

```
CALL IGCTRL(device,operat[,modif,...])
```

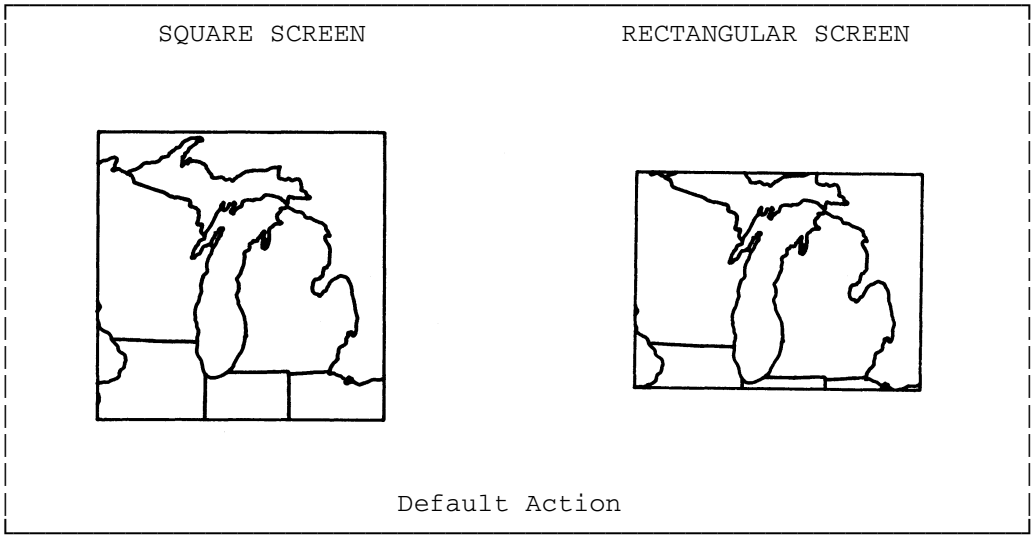
The "device" parameter may be either 'TERMINAL', 'CALCOMP', or a user-defined keyword for an auxiliary device (see the following section, "Auxiliary Devices"). The remaining parameters are passed directly to the associated device-dependent routine (DDR). The "operat" parameter is a keyword specifying the kind of control operation to be performed. Associated with each keyword is a fixed number of "modif" parameters that further define the control operation.

For a list of the control operations that apply to a specific device, see the description of that device in Appendix I. Control operations that do not apply to the current device are ignored without comment. Thus, IGCTRL calls that are intended for some other device will not cause errors.

Sense information about a device may be obtained by making a call to the subroutine AGSENS. For details, see the AGSENS description in Appendix B.

NONSQUARE SCREENS

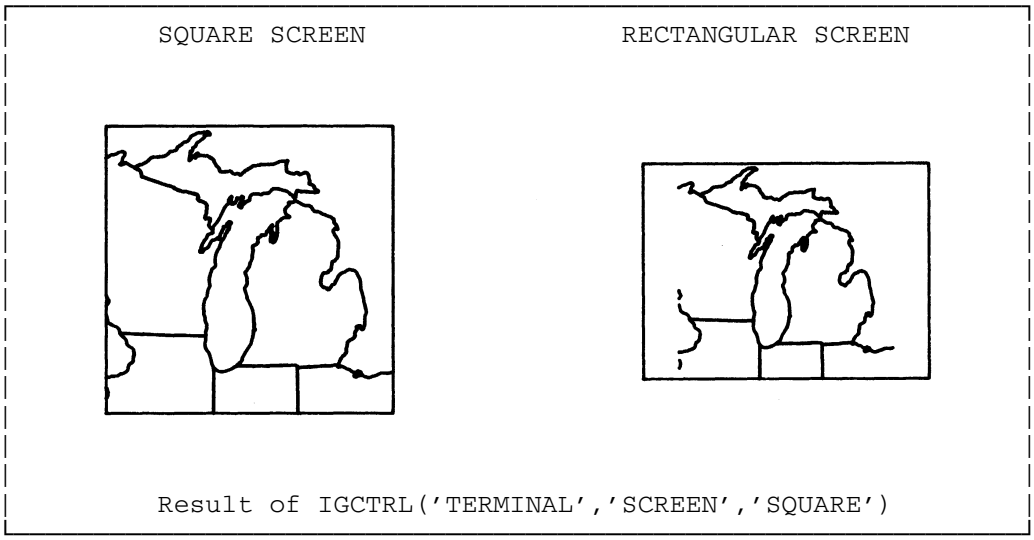
The most obvious device-to-device inconsistency is the absence of square screens. For nonsquare screens, the visible region is -1 to +1 in the X coordinate and -a to +a in the Y coordinate where a is the aspect ratio of the screen. This maintains the origin at the center of the screen regardless of its shape. If the screen is rectangular, however, the screen image will be truncated at the top and bottom:



The screen coordinates may be redefined so that they occupy the largest square centered within the screen, i.e., so that both the X and Y coordinates range from -1 to +1. This can be done by calling IGCTRL as follows:

```
CALL IGCTRL('TERMINAL','SCREEN','SQUARE')
```

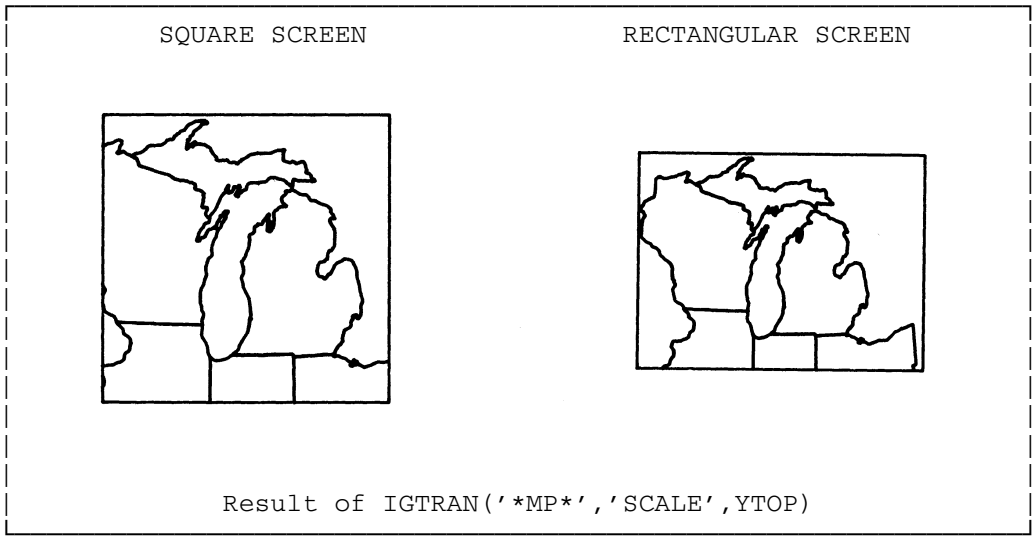
If the screen is rectangular, however, a strip on either side of the screen will be unused:



An alternative method for obtaining a square "screen" is to call AGSENS to obtain the aspect ratio of the terminal and then scale the main picture as follows:

```
YTOP=AGSENS('TERMINAL','YSIZE')
CALL IGTRAN('*MP*','SCALE',YTOP)
```

This shrinks the main picture by a factor of YTOP, which effectively makes the bottom and top of the screen -1 and +1, respectively, and the left and right of the screen -1/YTOP and +1/YTOP, respectively. As a result, the whole screen is utilized:



ERASING THE SCREEN

Graphics terminals utilize the screen for two purposes, graphic output and printed output. Before entering into textual dialogue with the user, the user program may erase the screen by calling IGCTRL as follows:

```
CALL IGCTRL('TERMINAL','ERASE')
```

This operation does not alter the IG data structure. Rather, it sends a command to the terminal to erase the screen image. Upon the next call to IGDRON('TERMINAL'), the screen image will be regenerated.

POSITIONING THE KEYBOARD CURSOR

Most storage-tube-type terminals have a keyboard cursor which indicates the point at which keyboard input is typed and program output is printed. This cursor may be placed at any position on the screen by calling IGCTRL as follows:

```
CALL IGCTRL('TERMINAL','POSN',X,Y)
```

X and Y are given in -1 to +1 screen coordinates. This operation is independent of the IG data structure.

SELECTIVE ERASE SIMULATION

Many storage-tube-type terminals have no selective erase facility. This means that individual lines cannot be erased from the screen; the whole screen must be erased at once. Thus, any calls to IGDRON ('TERMINAL') that require the erasure of individual lines will force the erasure of the entire screen and the regeneration of the entire image. This mode of operation is known as the automatic-erase mode.

It is sometimes preferable to use the keep mode of operation. In this mode, calls to IGDRON('TERMINAL') do not erase the screen. Furthermore, IGDRON does not regenerate the entire image but makes only the necessary additions. Thus, the screen image always consists of the current image superimposed on the old ones. Keep mode may be initiated by calling IGCTRL as follows:

```
CALL IGCTRL('TERMINAL','KEEP',1)
```

After the screen becomes overly cluttered with superimposed images, it may be erased and the current image regenerated by making the following calls:

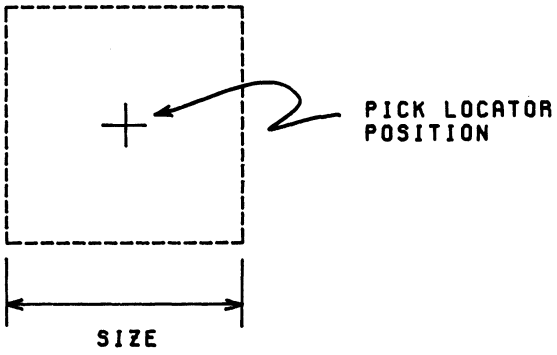
```
CALL IGCTRL('TERMINAL','ERASE')
CALL IGDRON('TERMINAL')
```

Automatic-erase mode may be restored by calling IGCTRL as follows:

```
CALL IGCTRL('TERMINAL','KEEP',0)
```

SETTING THE PICKBOX

In calls to IGPIKS, many terminals use a pickbox to determine which subpicture was hit by the pick operation. This works as follows. When the user enters the pick locator position, IGPIKS constructs an imaginary, square pickbox centered on the locator position:



By default, the size of the pickbox is 0.05, expressed as a fraction of the full -1 to +1 viewport (i.e., the size is 0.1 in the screen coordinate system). Using this pickbox as a kind of "viewport", IGPIKS simulates the regeneration of the screen image. If part of a subpicture is "displayed" in the pickbox, a hit is made on that subpicture. In the event that resolution between subpictures becomes a problem, it will be useful to reduce the size of the pickbox. This may be done by making the following call:

```
CALL IGCTRL('TERMINAL','PICKBOX',S)
```

This sets the size of the pickbox to S, expressed as a fraction of the full -1 to +1 viewport.

OPERATIONS ON CALCOMP PLOTS

Whenever a call is made to IGDRON('CALCOMP'), a plot description is generated and written on logical I/O unit 9. This logical I/O unit may be assigned in the $RUN command for the user program:

```
$RUN userprog+*IG 9=pdsfile
```

This specifies that plot descriptions are to be written in the file "pdsfile". Alternatively, the user program may make a call of the form

```
CALL IGCTRL('CALCOMP','PFILE','pdsfile ')
```

to specify that plot descriptions are to be written in "pdsfile". Note that the file name must be followed by a trailing blank.

Whenever IGDRON generates a plot description, it prints the message: PDS: PLOT DESCRIPTION GENERATION BEGINS. This message may be turned off (but not turned back on) by making a call of the form:

```
CALL IGCTRL('CALCOMP','PHDR',0)
```

By default, the plot will be scaled to fit an 8.5x11-inch sheet of plotter paper. The -1 to +1 visible region (corresponding to the terminal screen) will be mapped into a 7.5x7.5-inch square, with an X margin of 0.5 inches and a Y margin of 1.75 inches. Thus, a line of length 1.0 in the screen coordinate system will be 3.75 inches long in the plot. A different scale may be specified for future plots by making the following call:

```
CALL IGCTRL('CALCOMP','SIZE',S)
```

In future calls to IGDRON('CALCOMP'), the -1 to +1 visible region will be mapped into a square S inches on a side with the same margins as above: an X margin of 0.5 inches and a Y margin of 1.75 inches. Different margins may be specified for future plots by making the following calls:

```
CALL IGCTRL('CALCOMP','XMAR',XMARGN)
CALL IGCTRL('CALCOMP','YMAR',YMARGN)
```

The maximum size of the plotter paper is 360 inches in the X dimension and 33 inches in the Y dimension. If a plot is scaled larger than this, a portion of the visible region will be clipped off. The first parts to be sacrificed will be the blank margins, followed by the edges of the plot. The center of the plot will always be visible. For example, if the S value above is set to 40.0, the entire X dimension will be visible but the top and bottom 3.5 inches of the Y dimension will be clipped off.

AUXILIARY DEVICES

So far, the keywords 'TERMINAL', 'CALCOMP', and 'SAVE' have been used to refer to graphic I/O devices. The user may define additional keywords to refer to auxiliary devices. This may be done by calling the subroutine IGAUXD, as in the following example:

```
CALL IGAUXD('HARDCOPY','*IG.TX4662 ')
```

This defines the keyword 'HARDCOPY'. Such a keyword may not be 'TERMINAL', 'CALCOMP', 'SAVE', or an existing user-defined keyword. In addition, it must begin with an alphanumeric character. As with most IG keywords, if more than four letters are given, only the first four are used. Thus 'HARDCOPY' is equivalent to 'HARD'. The second parameter, '*IG.TX4662 ', gives the name of the file containing the appropriate device-dependent routine (DDR). This name must always be followed by a trailing blank. In this example, the DDR is for the Tektronix 4662 plotter. It is assumed that such a plotter is attached to the user's terminal. In general, calls to IGAUXD take the form

```
CALL IGAUXD(device,ddrfil)
```

where "device" is the user-defined keyword for the auxiliary device and "ddrfil" is the name of the file containing the appropriate DDR. This name will always be of the form *IG.devnam, where "devnam" is the name of the device. The legal file names are given in the following list:

'*IG.CCMP ' The DDR for the CalComp plotter. Output is automatically written on logical I/O unit 9 or a file or device specified by the user program. Note that this device may already be referenced as 'CALCOMP'.

'*IG.HP7203 ' The DDR for the Hewlett-Packard 7203A plotter. This auxiliary device is assumed to be connected to the user's terminal. The IGAUXD subroutine will prompt the user to confirm this.

'*IG.HP7221 ' The DDR for the Hewlett-Packard 7221A plotter. This auxiliary device is assumed to be connected to the user's terminal. The IGAUXD subroutine will prompt the user to confirm this.

'*IG.HIDP11 ' The DDR for the Houston Instruments Data Plotter 11. This auxiliary device is assumed to be connected to the user's terminal. The IGAUXD subroutine will prompt the user to confirm this.

'*IG.PRNT '	The printer-plot DDR. Output for this auxiliary device is automatically written on *MSINK*. Various printer characters are used to simulate the shapes of lines.
'*IG.SAVE '	The object-saving DDR. Output is automatically written on SPUNCH or a file or device specified by the user program. Note that the object-saving DDR may already be referenced as 'SAVE'.
'*IG.TTY '	The DDR for nongraphics terminals (e.g., Teletypes). Output for this auxiliary device is automatically sent to the user's terminal (or to the line printer in batch). However, instead of being graphic output, this takes the form of a text description of the changes that have been made to the IG data structure since the last call to IGDRON for this auxiliary device. This text description can be useful for debugging purposes. See "Nongraphics Terminals" in Appendix I.
'*IG.TX4662 '	The DDR for the Tektronix 4662 plotter. This auxiliary device is assumed to be connected to the user's terminal. The IGAUXD subroutine will prompt the user to confirm this.

If a file name is given that is not in the above list, an error message will be printed and MTS called. If a $RESTART is then given, the CalComp DDR ('*IG.CCMP ') will be substituted.

Once an auxiliary device has been established by calling IGAUXD, it may be referenced by the user-defined keyword. For example, output may be sent to the auxiliary device 'HARDCOPY' by making the following call to IGDRON:

```
CALL IGDRON('HARDCOPY')
```

Device-dependent information may be obtained and control operations performed by making calls such as the following:

```
CW = AGSENS('HARDCOPY','XCHR')

CALL IGCTRL('HARDCOPY','SCREEN','FULL')
```

The valid control operations for each device type are listed in the individual device descriptions in Appendix I.

APPENDIX A: SUMMARY INFORMATION

The following lists of subroutines are intended for quick reference. Each subroutine is described in detail in Appendix B.

ALPHABETICAL LISTING OF SUBROUTINES

```
   value = AGSENS(device,inftyp)

   value = DGSENS(device,inftyp)

|  CALL IGATTS(namsub[,attr,atval[,atval,...])
|
|  CALL IGATTB(attr,atval[,atval,...])

   CALL IGAUXD(device,ddrfil)

   CALL IGBGNO(namobj)

   intnam = IGBGNO(0)

   CALL IGBGNS(namsub[,transf,...])

   intnam = IGBGNS(0[,transf,...])

   CALL IGCTNS(namsub)

   CALL IGCTRL(device,operat[,modif,...])

   CALL IGDA(x,y[,z])

   CALL IGDELO(namobj)

   CALL IGDELS(namsub)

   CALL IGDR(dx,dy[,dz])

   CALL IGDRON(device)

   CALL IGENDO(namobj)

   CALL IGENDS(namsub)

   CALL IGFMT(var,format[,nchars[,ndec]])
```

```
   CALL IGFMTH(var,format[,nchars[,ndec]])
   value = IGINFO(name,inftyp[,val,...])
   CALL IGINIT
   CALL IGLIKE(namsub,attrib,frmsub[,attrib,frmsub,...])
   CALL IGLOAD(fdname[,numobj,list][,'PRES'])
   CALL IGMA(x,y[,z])
   CALL IGMR(dx,dy[,dz])
   CALL IGPDSW(isw,jsw)
   index = IGPIKC(cpic,x,y,name[,name,...])
   namsub = IGPIKN(level[,xhit,yhit])
   index = IGPIKS(name[,name,...])
   CALL IGPUTO(namobj[,namsub[,transf,...]])
   intnam = IGPUTO(namobj[,0[,transf,...]])
   CALL IGRNAM(oldnam,newnam)
|  CALL IGSAVE(namsub[,svname])
   value = IGSENS(device,inftyp)
   CALL IGSYM(char)
   CALL IGTRAN(namsub,type[,modif,...][,type[,modif,...],...])
   CALL IGTXT(string[,string,...])
   CALL IGTXTH(string)
   CALL IGUSER(namsub,putget,value)
   CALL IGVEC(nlines[,intvec[,movtyp[,nwords]]],xvec,yvec[,zvec])
   CALL IGVWPT(namsub,xl,xr,yb,yt)
   icode = IGXYIN(xin,yin)
```

LISTING OF SUBROUTINES BY FUNCTION

Picture Creation and Modification

```
CALL IGBGNS(namsub[,transf,...])

intnam = IGBGNS(0[,transf,...])

CALL IGCTNS(namsub)

CALL IGDELS(namsub)

CALL IGENDS(namsub)
```

Picture Attribute Specification

```
CALL IGATTS(namsub[,attr,atval[,atval,...])

CALL IGATTB(attr,atval[,atval,...])

CALL IGLIKE(namsub,attrib,frmsub[,attrib,frmsub,...])

CALL IGTRAN(namsub,type[,modif,...][,type[,modif,...],...])

CALL IGUSER(namsub,putget,value)

CALL IGVWPT(namsub,xl,xr,yb,yt)
```

Object Creation and Manipulation

```
CALL IGBGNO(namobj)

intnam = IGBGNO(0)

CALL IGDELO(namobj)

CALL IGENDO(namobj)

CALL IGPUTO(namobj[,namsub[,transf,...]])

intnam = IGPUTO(namobj[,0[,transf,...]])
```

Picture and Object Contents Specification

```
CALL IGDA(x,y[,z])
CALL IGDR(dx,dy[,dz])
CALL IGFMT(var,format[,nchars[,ndec]])
CALL IGFMTH(var,format[,nchars[,ndec]])
CALL IGMA(x,y[,z])
CALL IGMR(dx,dy[,dz])
CALL IGSYM(char)
CALL IGTXT(string[,string,...])
CALL IGTXTH(string)
CALL IGVEC(nlines[,intvec[,movtyp[,nwords]]],xvec,yvec[,zvec])
```

Graphic Input

```
index = IGPIKC(cpic,x,y,name[,name,...])
namsub = IGPIKN(level[,xhit,yhit])
index = IGPIKS(name[,name...])
icode = IGXYIN(xin,yin)
```

Device-Dependent Operations

```
value = AGSENS(device,inftyp)
value = DGSENS(device,inftyp)
CALL IGAUXD(device,ddrfil)
CALL IGCTRL(device,operat[,modif,...])
CALL IGDRON(device)
CALL IGSAVE(name[,svname])
value = IGSENS(device,inftyp)
```

Miscellaneous

```
value = IGINFO(name,inftyp[,val,...])

CALL IGINIT

CALL IGLOAD(fdname[,numobj,list][,'PRES'])

CALL IGPDSW(isw,jsw)

CALL IGRNAM(oldnam,newnam)
```

MTS 17: Integrated Graphics System

December 1980

APPENDIX B: SUBROUTINE CALLING SEQUENCES

This appendix gives the complete calling sequence for each subroutine in the *IG library and a capsule summary of what the subroutine does. All optional parameters are enclosed in square brackets [ ].

Many IG subroutines have variable-length parameter lists. The last parameter in such a list is distinguished by a leading 1-bit in its left-most byte. FORTRAN programmers need not worry about this bit since FORTRAN automatically turns it on. However, assembler-language programmers must explicitly turn this bit on. See the section "Calling Conventions" in MTS Volume 3, System Subroutine Descriptions.

AGSENS

Purpose: To obtain device-dependent information about a graphics device.

Calling Sequence:

FORTRAN: value = AGSENS(device,inftyp)

Parameters:

device is a keyword specifying the device. This may be one of the following:

'TERM' the current terminal
'CALC' the CalComp plotter

Device may also be a user-defined keyword for an auxiliary device. See the description of IGAUXD.

inftyp is a keyword specifying the kind of information to be obtained. This may be one of the following:

'YSIZ' the maximum displayable Y coordinate for the device.
'XSIZ' the maximum displayable X coordinate for the device.
'XCHR' the width, in -1 to +1 screen coordinates, of hardware characters generated by the device.
'NAME' the eight-character name of the device. (The name of the device is "devnam" if and only if the device-dependent routine resides in the file *IG.devnam. See Appendix I.)

Values Returned:

value is the value of the information.

Description: This subroutine obtains information from the device-dependent routine (DDR) associated with the specified device.

DGSENS

Purpose: To obtain device-dependent information about a graphics device. This is an alternate entry name for AGSENS. See the AGSENS description.

IGATTB

| Purpose: To dynamically change the attributes in effect. The changed attributes take effect immediately and the changes in the attributes are not inherited by the lower-level subpictures. This is a way for example to have two different colors in a single subpicture.

Calling Sequence:

FORTRAN: CALL IGATTB(attr,atval[,atval,...])

Parameters:

attr is a keyword describing the attribute to be specified. The legal values are listed below.

atval are additional parameters that specify the attribute.

Description: Each attribute keyword attr must be followed by a fixed number of atval values. The legal attribute values are given in the following list:

'PEN ',pnum — The pen number, pnum, may be any integer from 0 to 32767. This is a code that represents various line qualities (e.g., hue and intensity) that can be produced by the output device (using various "pens"). The actual interpretation of a given pen number will depend on the specific output device.

| 'PFILL',fillnum — The polygon fill number fillnum may be any integer from 0 to 255. This is a number that represents different fill patterns to be used in filling polygons. The actual interpretation of a given fill number will depend on the specific output device.

| 'PEDGE',edgenum — The polygon edge number edgenum may be any integer from 0 to 255. This is a number that represents the different line qualities (e.g., hue and intensity) to be used for the edges of the polygon. The actual interpretation of a given polygon edge number will depend on the specific output device.

'TSCALE',tscl — The default text scale, tscl, may be any positive floating-point number. This is the default scale (in subpicture coordinates) for text added by IGTXT or IGFMT.

'FONT',charset — The default character set or font, charset, may be one of the following:

'STANDARD'	uppercase ASCII
'7ASCII'	full 7-bit ASCII
'SANSERIF.1'	
'SANSERIF.2'	
'SANSERIF.CART'	
'ROMAN.2A'	
'ROMAN.2'	
'ROMAN.3'	
'ITALIC.2A'	
'ITALIC.2'	
'ITALIC.3'	
'SCRIPT.1'	
'SCRIPT.2'	
'GREEK.1'	
'GREEK.2A'	
'GREEK.2'	
'GREEK.CART'	
'GREEK'	
'GOTHIC.ENGLISH'	
'GOTHIC.FRAKTUR'	
'GOTHIC.ITALIAN'	
'CYRILLIC.2'	

This is the default character set or font for text added by IGTXT, IGFMT, or IGSYM. These keywords must be spelled out in full. (This is an exception to the rule that, with most IG keywords, the first four letters are sufficient.)

'TPLANE',dx,dy,dz — The text plane is described by the floating-point values dx, dy, and dz. Text strings for the subpicture are drawn along the same direction as the vector from the origin to (dx,dy,dz).

IGATTS

Purpose: To specify attributes for a subpicture.

Calling Sequence:

FORTRAN: CALL IGATTS(namsub[,attr,atval[,atval,...])

Parameters:

namsub is the subpicture for which attributes are to be specified.

attr is a keyword describing the attribute to be specified. The legal values are listed below.

atval are additional parameters that specify the attribute.

Description: Each attribute keyword attr must be followed by a fixed number of atval values. The legal attribute values are given in the following list:

'PEN ',pnum — The pen number, pnum, may be any integer from 0 to 32767. This is a code that represents various line qualities (e.g., hue and intensity) that can be produced by the output device (using various "pens"). The actual interpretation of a given pen number will depend on the specific output device.

'PFILL',fillnum — The polygon fill number fillnum may be any integer from 0 to 255. This is a number that represents different fill patterns to be used in filling polygons. The actual interpretation of a given fill number will depend on the specific output device.

'PEDGE',edgenum — The polygon edge number edgenum may be any integer from 0 to 255. This is a number that represents the different line qualities (e.g., hue and intensity) to be used for the edges of the polygon. The actual interpretation of a given polygon edge number will depend on the specific output device.

'TSCALE',tscl — The default text scale, tscl, may be any positive floating-point number. This is the default scale (in subpicture coordinates) for text added by IGTXT or IGFMT.

'FONT',charset — The default character set or font, charset, may be one of the following:

'STANDARD'	uppercase ASCII
'7ASCII'	full 7-bit ASCII
'SANSERIF.1'	
'SANSERIF.2'	
'SANSERIF.CART'	
'ROMAN.2A'	
'ROMAN.2'	
'ROMAN.3'	
'ITALIC.2A'	
'ITALIC.2'	
'ITALIC.3'	
'SCRIPT.1'	
'SCRIPT.2'	
'GREEK.1'	
'GREEK.2A'	
'GREEK.2'	
'GREEK.CART'	
'GREEK'	
'GOTHIC.ENGLISH'	
'GOTHIC.FRAKTUR'	
'GOTHIC.ITALIAN'	
'CYRILLIC.2'	

This is the default character set or font for text added by IGTXT, IGFMT, or IGSYM. These keywords must be spelled out in full. (This is an exception to the rule that, with most IG keywords, the first four letters are sufficient.)

'TPLANE',dx,dy,dz — The text plane is described by the floating-point values dx, dy, and dz. Text strings for the subpicture are drawn along the same direction as the vector from the origin to (dx,dy,dz).

IGAUXD

Purpose: To assign a keyword to an auxiliary device and specify the device-dependent routine (DDR) to be used for the device.

Calling Sequence:

FORTRAN: CALL IGAUXD(device,ddrfil)

Parameters:

device is the keyword (four characters) to be assigned to the auxiliary device. As with most IG keywords, if more than four characters are given, only the first four will be used.

ddrfil is the name of the file where the appropriate DDR resides. This name must be followed by a blank. The legal values are listed below.

Description: The first parameter device may not be 'TERM', 'CALC', or 'SAVE', and may not already be in use as a keyword for an auxiliary device. Otherwise, an error message will be printed and MTS called. The second parameter ddrfil must be one of the following:

'*IG.CCMP '	the DDR for the CalComp plotter
'*IG.HP7203 '	the DDR for the Hewlett-Packard 7203A plotter
'*IG.HP7221 '	the DDR for the Hewlett-Packard 7221A plotter
'*IG.HPGL '	the DSR for the Hewlett-Packard HPGL Plotters (7220A, 7220C, 7220T, 7470, 7475, and 9872A)
'*IG.HIDP11 '	the DDR for the Houston Instruments Data Plotter 11
'*IG.PRNT '	the printer-plot DDR
'*IG.SAVE '	the object-saving DDR
'*IG.TTY '	the DDR for nongraphics terminals
'*IG.TX4662 '	the DDR for the Tektronix 4662 plotter

If ddrfil is invalid, an error message will be printed and MTS called. If a $RESTART is then given, the CalComp DDR ('*IG.CCMP ') will be used. If ddrfil refers to one of the remote plotters, the user will be prompted to confirm the presence of such a device. If the user then answers in the negative, the CalComp DDR will be substituted.

Example: CALL IGAUXD('HARDCOPY','*IG.TX4662 ')

IGBGNO

Purpose: To create a new object or empty and reactivate an existing object.

Calling Sequence:

FORTRAN: CALL IGBGNO(namobj)

intnam = IGBGNO(0)

Parameters:

namobj This may be one of the following:

(1) A (currently unused) name to be assigned to a new object. This must consist of four characters and must begin with a letter.
(2) The name (four-character or internal) of an existing object.

Values Returned:

intnam is a unique, internal, object name generated by IG. This name can be used in the same way as a user-specified object name.

Description: This subroutine pushes the name of the active subpicture/object onto the stack of open subpictures/objects. (A subpicture/object is open if it has not been terminated by IGENDS/IGENDO.) If namobj does not already exist, it is created. If namobj does already exist, all of its lines, text, and subpictures are deleted. Then, namobj becomes the active object. The current position for namobj is reset to the origin (0,0). Lines and text may then be added by calling picture-description subroutines.

IGBGNS

Purpose: To create a new subpicture or empty and reactivate an existing subpicture.

Calling Sequence:

FORTRAN: CALL IGBGNS(namsub[,transf,...])

intnam = IGBGNS(0[,transf,...])

Parameters:

namsub This may be one of the following:

(1) A (currently unused) name to be assigned to a new subpicture. This must consist of four characters and must begin with a letter.
(2) The name (four-character or internal) of an existing subpicture.

transf is an optional parameter list that may be used to specify the transformation associated with the subpicture. These parameters have the same format as the second through last parameters of IGTRAN.

Values Returned:

intnam is a unique, internal, subpicture name generated by IG. This name can be used in the same way as a user-specified subpicture name.

Description: This subroutine pushes the name of the active subpicture/object onto the stack of open subpictures/objects. (A subpicture/object is open if it has not been terminated by IGENDS/IGENDO.) If namsub does not already exist, it is created as a subpicture of the active subpicture/object. If namsub does already exist, all of its lines, text, and subpictures are deleted. Then, namsub becomes the active subpicture. The current position for namsub is reset to the origin (0,0). Lines and text may then be added by calling picture-description subroutines. If the transf parameter list is given, a transformation is specified for namsub.

Example:

```
      DO 10 I = 1,10
      CALL IGBGNS('PICT')
            .
            .
      CALL IGENDS('PICT')
      CALL IGDRON('TERM')
10    CONTINUE
```

On the first iteration, 'PICT' is created. On each subsequent iteration, 'PICT' is respecified.

IGCTNS

Purpose: To reactivate an existing subpicture without deleting its current contents.

Calling Sequence:

FORTRAN: CALL IGCTNS(namsub)

Parameters:

namsub is the name (four-character or internal) of the subpicture to be reactivated.

Description: This subroutine pushes the name of the active subpicture/object onto the stack of open subpictures/objects. (A subpicture/object is open if it has not been terminated by IGENDS/IGENDO.) Then, namsub becomes the active subpicture. The current position for namsub remains the same as when namsub was last active. Additional lines and text may then be added by calling picture-description subroutines.

IGCTRL

Purpose: To perform a device-dependent control operation on a device.

Calling Sequence:

FORTRAN: CALL IGCTRL(device,operat[,modif,...])

Parameters:

device is a keyword specifying the device. This may be one of the following:

'TERM' - the current terminal
'CALC' - the CalComp plotter
'SAVE' - a file or other storage device

Device may also be a user-defined keyword for an auxiliary device. See the description of IGAUXD.

operat is a keyword specifying the operation to be performed on the device. The legal operations depend on the device type and are listed in the individual device descriptions in Appendix I. The legal operations for 'SAVE' are listed in the "Object-Saving Files or Devices" description in Appendix I.

modif are additional parameters necessary for certain values of operat.

Description: The second through last parameters are passed to the associated device-dependent routine (DDR), which either performs an immediate operation (such as erasing the screen) or sets a switch that affects future IGDRON operations. If the specified operation is not recognized by the DDR, it is ignored without comment.

Examples:

```
CALL IGCTRL('TERM','ERASE')
CALL IGCTRL('TERM','POSN',-1.,0.)
```

IGDA

Purpose: To add a visible line (one with a "draw" flag) to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGDA(x,y[,z])

Parameters:

x,y,z are the absolute coordinates, in the coordinate space of the active subpicture/object, of the endpoint of the new line. If omitted, z has a value equal to the current Z coordinate.

Description: The line is drawn from the current position to the absolute point (x,y,z). This point then becomes the new current position.

IGDELO

Purpose: To delete an object from the internal data structure.

Calling Sequence:

FORTRAN: CALL IGDELO(namobj)

Parameters:

namobj is the name (four-character or internal) of the object to be deleted.

Description: This subroutine first deletes all instances of namobj, i.e., all subpictures created by a call to IGPUTO with namobj as the first parameter. It then deletes the object namobj.

All subpictures of namobj are also deleted. However, any user-defined objects linked into the substructure of namobj are not deleted.

Namobj should not be active or on the stack of open subpictures/objects at the time this subroutine is called.

IGDELS

Purpose: To delete a subpicture from the internal data structure.

Calling Sequence:

FORTRAN: CALL IGDELS(namsub)

Parameters:

namsub is the name (four-character or internal) of the subpicture to be deleted.

Description: This subroutine deletes the subpicture namsub.

All subpictures of namsub are also deleted. However, any user-defined objects linked into the substructure of namsub are not deleted.

Namsub should not be active or on the stack of open subpictures/objects at the time this subroutine is called.

IGDR

Purpose: To add a visible line (one with a "draw" flag) to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGDR(dx,dy[,dz])

Parameters:

dx,dy,dz are the relative displacements from the current position, in the coordinate space of the active subpicture/object, of the endpoint of the new line. If omitted, dz has a value of zero.

Description: The line is drawn with the displacement (dx,dy,dz) relative to the current position. The new endpoint then becomes the new current position.

IGDRON

Purpose: To display the main picture and all subpictures on the terminal or some other device.

Calling Sequence:

FORTRAN: CALL IGDRON(device)

Parameters:

device is a keyword specifying the device on which the picture is to be displayed. This may be one of the following:

'TERM' the current terminal
'CALC' the CalComp plotter
'SAVE' a file or other storage device

Device may also be a user-defined keyword for an auxiliary device. See the description of IGAUXD.

Description: This subroutine scans the internal data structure and translates it into the hardware commands necessary to generate the screen image or plot. IGDRON writes these commands to the terminal (for 'TERM'), logical I/O unit 9 (for 'CALC'), or a file or other storage device (for 'SAVE'). IGDRON and IGSAVE are the only subroutines that transmit hardware commands to output devices. IGDRON is described in more detail in Appendices H and I.

IGENDO

Purpose: To terminate an object and reactivate the subpicture/object that was active at the time of the matching call to IGBGNO.

Calling Sequence:

FORTRAN: CALL IGENDO(namobj)

Parameters:

namobj is the name (four-character or internal) of the active object.

Description: This subroutine first determines if namobj is the active object. (This is an error check to ensure that IGBGNO and IGENDO remain in phase.) If so, it terminates namobj and pops the name of the previously active subpicture/object from the stack of open subpictures/objects. It then reactivates the previously active subpicture/object.

IGENDS

Purpose: To terminate a subpicture and reactivate the subpicture/object that was active at the time of the matching call to IGBGNS or IGCTNS.

Calling Sequence:

FORTRAN: CALL IGENDS(namsub)

namsub is the name (four-character or internal) of the active subpicture.

Description: This subroutine first determines if the namsub is the active subpicture. (This is an error check to ensure that IGBGNS and IGENDS remain in phase.) If so, it terminates namsub and pops the name of the previously active subpicture/object from the stack of open subpictures/objects. It then reactivates the previously active subpicture/object.

IGFMT

Purpose: To convert a variable value to character form and add the resulting transformable text string to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGFMT(var,format[,nchars[,ndec]])

Parameters:

var is the variable to be converted to a text string.

format is a code for the conversion:

'A'	character string
'F'	REAL*4
'I'	INTEGER*4
'E'	REAL*4 (exponential notation)
'D'	REAL*8 (exponential notation)
'H'	INTEGER*2

nchars is the number of characters (counting blanks) to be included in the text string. If nchars is omitted, the text string will be just long enough to contain the converted variable (except when the format is A, in which case the text string will be of length 1).

ndec is the number of digits to the right of the decimal point in formats F, E, and D. If ndec is omitted, the default value nchars/2 will be used.

Description: This subroutine converts the variable to character form and adds the resulting text string to the active subpicture/object. The conversion format is specified in a manner similar to FORTRAN formats. The resulting text string will have the same scale and font as text strings produced by IGTXT. In addition, the resulting text string will be subject to any transformations that are applied to the subpicture.

IGFMTH

Purpose: To convert a variable value to character form and add the resulting hardware text string to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGFMTH(var,format[,nchars[,ndec]])

Parameters:

var is the variable to be converted to a text string.

format is a code for the conversion:

```
'A'   character string
'F'   REAL*4
'I'   INTEGER*4
'E'   REAL*4   (exponential notation)
'D'   REAL*8   (exponential notation)
'H'   INTEGER*2
```

nchars is the number of characters (counting blanks) to be included in the text string. If nchars is omitted, the text string will be just long enough to contain the converted variable (except when the format is A, in which case the text string will be of length 1).

ndec is the number of digits to the right of the decimal point in formats F, E, and D. If ndec is omitted, the default value nchars/2 will be used.

Description: This subroutine converts the variable to character form and adds the resulting hardware text string to the active subpicture/object. The conversion format is specified in a manner similar to FORTRAN formats. The resulting text string will be treated like a text string produced by IGTXTH.

IGINFO

Purpose: To obtain information about a subpicture or object.

Calling Sequence:

FORTRAN: value = IGINFO(name,inftyp[,val,...])

Parameters:

name is the name (four-character or internal) of the subpicture or object for which information is to be returned.

inftyp is a keyword specifying the kind of information to be returned. The legal values are listed below.

Values Returned:

value is a value returned by the subroutine. The interpretation of this value depends on the inftyp keyword.

val are additional values returned by the subroutine. The interpretation of these values depends on the inftyp keyword.

Description: This subroutine returns information about the specified subpicture or object. The kind of information is determined by the inftyp keyword, which must be followed by some fixed number of val parameters, as in the following list:

'INTERNAL' If the subpicture or object exists, its internal name is returned in value. Otherwise, a zero is returned.

'EXTERNAL' If the subpicture or object exists and has a user-specified, four-character, external name, this name is returned in value. Otherwise, a zero is returned.

'OBJECT' If name is a subpicture that has an object defined by IGBGNO, the internal name of the object is returned in value. If name is an object defined by IGBGNO, its internal name is returned in value. Otherwise, a zero is returned.

'OPEN' If the subpicture or object is open

(i.e., if IGBGNS/IGCTNS or IGBGNO has been called for the subpicture or object but IGENDS or IGENDO has not been called), a nonzero value is returned in value. Otherwise, a zero is returned.

'VWPT',xl,xr,yb,yt — If name is a subpicture, its viewport parameters are returned in xl,xr,yb, yt. These parameters are the same as the second through last parameters to IGVWPT. If these parameters are successfully returned, a nonzero value is returned in value. Otherwise, a zero is returned.

'CURRENT',x,y[,z] — The current position for the subpicture or object is returned in x,y,z. If these parameters are successfully returned, a nonzero value is returned in value. Otherwise, a zero is returned.

'MINIMUM',x,y[,z] — The minimum X, Y, and Z coordinates for the subpicture or object are returned in x,y,z. If these parameters are successfully returned, a nonzero value is returned in value. Otherwise, a zero is returned.

'MAXIMUM',x,y[,z] — The maximum X, Y, and Z coordinates for the subpicture or object are returned in x,y,z. If these parameters are successfully returned, a nonzero value is returned in value. Otherwise, a zero is returned.

'TSCALE',tscl — The text scale for the object or subpicture is returned in tscl. If the value is successfully obtained, a nonzero value is returned in value; otherwise, a zero is returned.

'TPLANE',dx,dy,dz — The values for the text plane are returned in dx, dy, and dz. These values are the same as the values specified in the last call to IGATTS or IGATTB for the text plane of name. If the value is successfully obtained, a nonzero value is returned in value; otherwise, a zero is returned.

'TXTLENGTH',string,length,xl,xr — The values xr and xl should be used with the text scale to calculate the actual length of string, a text string with length number of characters. The values returned in xl and xr are in text units and should not be used as absolute coordinates. The actual

length is calculated by first getting the text scale in tscl

```
value = IGINFO(name,'TSCALE',tscl)
```

then getting xl and xr

```
value = IGINFO(name,'TXTLENGTH',
              string,length,xl,xr)
```

and finally computing (xr-xl)*tscl to obtain the actual length.

| Note that this example will not work if the default font or text scale are being used.

As with most IG keywords, if more than four characters are given, only the first four will be used.

Example:

```
      IVAL = IGINFO(NAMPIC,'MINI',XMIN,YMIN)
      IF(IVAL.EQ.0) GO TO 10
      IVAL = IGINFO(NAMPIC,'MAXI',XMAX,YMAX)
      IF(IVAL.EQ.0) GO TO 10
      CALL IGTRAN(NAMPIC,'WIND',XMIN,XMAX,YMIN,YMAX)
   10 CONTINUE
```

Page Revised September 1984 December 1980

IGINIT

Purpose: To initialize the IG system.

Calling Sequence:

FORTRAN: CALL IGINIT

Description: This subroutine must be called before any other IG subroutines may be called. It creates the main picture '*MP*' and makes it the active picture. All subpictures created by (nonnested) calls to IGBGNS or IGPUTO will then be subpictures of '*MP*'.

IGLIKE

Purpose: To specify attributes for a subpicture by copying them from other subpictures.

Calling Sequence:

FORTRAN: CALL IGLIKE(namsub[,attrib,frmsub,...])

Parameters:

namsub is the subpicture for which attributes are to be specified.

attrib is a keyword describing one or more attributes to be copied. This may be any of the following:

'TRAN' or 'XFRM'	transformation
'VWPT' or 'VIEW'	viewport
'PEN '	pen number
'TSCALE'	default text scale
'FONT'	default character set or font
'EVERYTHING'	transformation, viewport, pen number, default text scale, and default character set or font

frmsub is the subpicture from which the attribute is to be copied.

Description: Each attribute keyword attrib must be followed by the name of a single subpicture frmsub from which the attribute is to be copied. IGLIKE can be called at any time, whether namsub is active or not.

Example:

```
CALL IGLIKE('PIC1','TRAN','PIC2')
```

IGLOAD

Purpose: To reload a set of objects from a file or device.

Calling Sequence:

FORTRAN: CALL IGLOAD(fdname[,numobj,list][,modif])

Parameters:

fdname is the name of the file or device from which the objects are to be reloaded. This name must be followed by a trailing blank.

modif may be one of the following keywords:

'PRESERVE'	preserve the names of subpictures within the objects as they are reloaded.
'NOPRESERVE'	do not preserve the names of subpictures within the objects as they are reloaded.

Values Returned:

numobj is the number of objects that were reloaded. If numobj is specified, list must also be specified.

list is an array which contains the names of the objects that were reloaded. If list is specified, numobj must also be specified.

Description: The file or device fdname is assumed to contain a set of objects which were saved by calling IGDRON('SAVE') (see the section "Saving the Current Data Structure as an Object"). IGLOAD reads from fdname and reloads all objects it finds there, until it encounters an end-of-file. A reloaded object may replace an existing object (if it has the same name as an existing object) or become a new object. In the latter case, the new object may be attached to the data structure by calling IGPUTO.

The number of reloaded objects is returned in numobj and a list of their names is returned in list.

If modif is omitted or 'NOPRESERVE' is specified, the names of reloaded subpictures (i.e., subpictures of reloaded objects) will be changed to arbitrary names.

If 'PRESERVE' is specified, the names of reloaded subpictures will be preserved. If a reloaded subpicture has the same name as an existing subpicture, the contents of the existing subpicture will be replaced by the contents of the reloaded subpicture. Furthermore, the reloaded subpicture will be deleted from the reloaded object. Thus, the data structure may be altered significantly.

Example:

```
CALL IGLOAD('OBJECTFILE1 ')
```

IGMA

Purpose: To add an invisible line (one with a "move" flag) to the active subpicture/object. In other words, to move the current position.

Calling Sequence:

FORTRAN: CALL IGMA(x,y[,z])

Parameters:

x,y,z are the absolute coordinates, in the coordinate space of the active subpicture/object, of the endpoint of the move. If omitted, z has a value equal to the current Z coordinate.

Description: The move is made from the current position to the absolute point (x,y,z). This point then becomes the new current position.

IGMR

Purpose: To add an invisible line (one with a "move" flag) to the active subpicture/object. In other words, to move the current position.

Calling Sequence:

FORTRAN: CALL IGMR(dx,dy[,dz])

Parameters:

dx,dy,dz are the relative displacements from the current position, in the coordinate space of the active subpicture/object, of the endpoint of the move. If omitted, dz has a value of zero.

Description: The move is made with the displacement (dx,dy,dz) relative to the current position. The new endpoint then becomes the new current position.

IGPDSW

Purpose: To alter the default action of the IG-PDS interface.

Calling Sequence:

FORTRAN: CALL IGPDSW(isw,jsw)

Parameters:

isw is a switch that controls the operation of the IG versions of PLTBGN and PLTEND. If isw is 1 (the default), PLTBGN implicitly calls IGINIT (if necessary) and IGBGNS('PLOT'). Then PLTEND implicitly calls IGENDS('PLOT'), rescales the plot to fit within the screen boundaries, and calls IGDRON('TERMINAL'). If isw is 0, PLTBGN and PLTEND do not have any function.

jsw is a switch that controls the operation of the IG version of PLTEND. If jsw is 1 (the default), PLTEND prompts the user with "Blow-up, Redraw, Plot, or Continue?" and, depending on the reply, rescales the plot and calls IGDRON('TERMINAL'). If jsw is 0, PLTEND does not prompt the user.

Description: IG has its own versions of the PDS subroutines PLTBGN and PLTEND. The IGPDSW subroutine sets switches in the these subroutines to alter their operation (see Appendix J).

IGPIKC

Purpose: To pick a subpicture from a list of subpictures, using program-generated coordinates.

Calling Sequence:

FORTRAN: index = IGPIKC(cpic,x,y,name[,name,...])

Parameters:

cpic is the name (four-character or internal) of a subpicture.

x,y are coordinates in the coordinate space of cpic.

name is the name (four-character or internal) of a subpicture. This is the first element in a list of subpictures, from which one is to be picked.

Values Returned:

index is a number (in the list of subpicture names) of a subpicture that was picked, or zero, if no subpicture was picked.

Description: This subroutine implicitly calls IGDRON('TERMINAL') to update the screen image. It then maps the coordinates x and y into screen coordinates, using the transformation and viewport associated with cpic. The resulting screen coordinates represent an imaginary "pick locator" that is used to pick a subpicture, as if IGPIKS were called. Only those subpictures listed as the third through last parameters may be picked. If one of these is picked, the subroutine returns its number in the list. If none of these is picked, the subroutine returns the number zero.

Note that additional information resulting from a call to IGPIKC may be obtained by making calls to IGPIKN.

Example:

```
      1   CALL IGXYIN(X,Y)
          JUMP = IGPIKC('*AP*',X,Y,'BUT1','BUT2','BUT3')
          GO TO (100,200,300), JUMP
          WRITE(6,10)
     10   FORMAT('MISSED TRY AGAIN')
          GO TO 1
```

IGPIKN

Purpose: To obtain additional information about the most recent call to IGPIKC or IGPIKS.

Calling Sequence:

FORTRAN: namsub = IGPIKN(level[,xhit,yhit])

Parameters:

level is the nesting level for which information is to be obtained.

Values Returned:

namsub is the name (four-character or internal) of a subpicture, on level, within which the hit occurred. If no hit occurred on level, the value zero is returned.

xhit,yhit

are the coordinates, in the coordinate space of namsub, of the hit.

Description: Each subpicture in the IG data structure has a nesting level. Subpictures of the main picture '*MP*' have the nesting level 0, subpictures of these subpictures have the nesting level 1, and so on. Successively higher nesting-level numbers represent successively lower nesting levels.

On a call to IGPIKS or IGPIKC, a picture is picked if any of its subpictures (on any nesting level) is hit by the pick locator. A hit on a subpicture is considered to be a hit on all higher-level pictures containing the subpicture. The IGPIKN subroutine returns information, for level, about the hit. If no subpicture on level was hit, the value zero is returned and the remaining parameters are not changed. If a subpicture on level was hit, the name of the subpicture is returned in namsub. In addition, the coordinates of the hit, in the coordinate system of namsub, are returned in xhit and yhit.

If the coordinates xhit and yhit are not needed, they should be omitted. This will allow IGPIKN to execute much faster.

Note that repeated calls to IGPIKN at successively lower (numerically higher) nesting levels will eventually determine the lowest-level subpicture that was hit.

Note also that IGPIKN only returns information about the most recent call to either IGPIKS or IGPIKC.

IGPIKS

Purpose: To allow the user to pick a subpicture from a list of subpictures, using the pick locator.

Calling Sequence:

FORTRAN: index = IGPIKS(name[,name,...])

Parameters:

name is the name (four-character or internal) of a subpicture. This is the first element in a list of subpictures, from which one is to be picked.

Values Returned:

index is a number (in the list of subpicture names) of a subpicture that was picked, or zero, if no subpicture was picked.

Description: This subroutine implicitly calls IGDRON('TERMINAL') to update the screen image. It then waits for the user to pick a subpicture by hitting it with the pick locator. (This may be a light pen, cursor, or some other mechanism. The exact details depend on the terminal type. See Appendix I.) Only those subpictures listed as parameters may be picked. If one of these is picked, the subroutine returns its number in the list. If no subpicture in the list is picked, the subroutine will wait for the user to pick again.

Note that additional information resulting from a call to IGPIKS may be obtained by making calls to IGPIKN.

Example:

```
JUMP = IGPIKS('BUT1','BUT2','BUT3')
GO TO (100,200,300), JUMP
```

IGPOL2

Purpose: To add a two-dimensional polygon to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGPOL2(npnts,xarray,yarray)
CALL IGPOL2(npnts,nwords,xarray,yarray)

Parameters:

npnts is the number of points in the polygon, i.e., the number of values to extract from the arrays xarray and yarray for the vertices.

nwords is the number of fullwords separating the values to be extracted from the arrays xarray and yarray. If omitted, a value of 1 is assumed.

xarray,yarray
are REAL*4 arrays containing the coordinates of the vertices for the polygon. The first and last vertices of the polygon need not be the same.

Description: The polygon is placed in the active subpicture/object with its vertices located at the points given by the values in the arrays xarray and yarray. The new current position will be located at the position of the initial vertex. The polygon will be drawn using the values for the PFILL and PEDGE attributes which can be set using the routines IGATTS or IGATTB. The PFILL attribute specifies which polygon fill pattern will be used for the polygon, and the PEDGE attribute specifies which pen number will be used for the edges of the polygon.

Whenever possible, the polygon and its filling will be generated by hardware. If the device does not support polygon fill, then software simulation will be used.

The actual fill patterns used for the polygons will vary depending upon the actual device being used, but in general if the device has N colors, then the first N polygon fills should be the solid colors.

The software fills use PFILL numbers 0-8. PFILL number 0 will cause no filling to be done, numbers 1-4 will cause horizontal fill lines, with 1 being solid and 4 having the

| widest spacing between the lines. PFILL numbers 5-8 are
| the same as numbers 1-4 except that they use vertical
| lines. The PEDGE number will be used by the device as the
| pen number when the polygon edge is drawn. The interior
| fill lines are drawn using the current PEN number.

IGPOL3

Purpose: To add a three-dimensional polygon to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGPOL3(npnts,xarray,yarray,zarray)
CALL IGPOL3(npnts,nwords,xarray,yarray,zarray)

Parameters:

npnts is the number of points in the polygon, i.e., the number of values to extract from the arrays xarray, yarray, and zarray for the vertices.

nwords is the number of fullwords separating the values to be extracted from the arrays xarray, yarray, and zarray. If omitted, a value of 1 is assumed.

xarray,yarray,zarray
are REAL*4 arrays containing the coordinates of the vertices for the polygon. The first and last vertices of the polygon need not be the same.

Description: The polygon is placed in the active subpicture/object with its vertices located at the points given by the values in the arrays xarray, yarray, and zarray. The new current position will be located at the position of the initial vertex. The polygon will be drawn using the values for the PFILL and PEDGE attributes which can be set using the routines IGATTS or IGATTB. The PFILL attribute specifies which polygon fill pattern will be used for the polygon, and the PEDGE attribute specifies which pen number will be used for the edges of the polygon.

Whenever possible, the polygon and its filling will be generated by hardware. If the device does not support polygon fill, then software simulation will be used.

The actual fill patterns used for the polygons will vary depending upon the actual device being used, but in general if the device has N colors, then the first N polygon fills should be the solid colors.

The software fills use PFILL numbers 0-8. PFILL number 0 will cause no filling to be done, numbers 1-4 will cause

horizontal fill lines, with 1 being solid and 4 having the widest spacing between the lines. PFILL numbers 5-8 are the same as numbers 1-4 except that they use vertical lines. The PEDGE number will be used by the device as the pen number when the polygon edge is drawn. The interior fill lines are drawn using the current PEN number.

IGPUTO

Purpose: To create a new subpicture or respecify an existing subpicture, as an instance of a user-defined object.

Calling Sequence:

FORTRAN: CALL IGPUTO(namobj[,namsub[,transf,...]])

intnam = IGPUTO(namobj[,0[,transf,...]])

Parameters:

namobj is the name (four-character or internal) of the object to be reproduced.

namsub If present, this must be one of the following:

(1) A (currently unused) name to be assigned to a new subpicture. This must consist of four characters and must begin with a letter.
(2) The name (four-character or internal) of an existing subpicture.

If omitted, a unique, internal, subpicture name will be generated by IG.

transf is an optional parameter list that may be used to specify the transformation associated with the subpicture. These parameters have the same format as the second through last parameters of IGTRAN.

intnam is a unique, internal, subpicture name generated by IG. This name can be used in the same way as a user-specified subpicture name.

Description: If namsub does not already exist, it is created as a subpicture of the active picture. If namsub does already exist, all of its lines, text, and subpictures are deleted. Then namsub is made an instance of the object namobj, i.e., its contents are defined as a copy of the contents of namobj. If the transf parameter list is given, a transformation is specified for namsub.

December 1980

Examples:

```
CALL IGPUTO('OBJT')
CALL IGPUTO('OBJT','INST')
INTNAM = IGPUTO('OBJT',0,'MOVE'.1.,.2)
```

IGRNAM

Purpose: To rename an existing subpicture or object.

Calling Sequence:

FORTRAN: CALL IGRNAM(oldnam,newnam)

Parameters:

oldnam is the name (four-character or internal) of the subpicture or object to be renamed.

newnam is the new four-character name to be assigned to the subpicture or object. This name must begin with a letter.

Description: This subroutine changes the name of the specified subpicture or object. It cannot, however, be used to change the name of the main picture '*MP*'.

IGSAVE

Purpose: To save the definition of an object or subpicture.

Calling Sequence:

FORTRAN: CALL IGSAVE(name[,svname])

Parameters:

name is the name of the object or subpicture to be saved.

svname is an alternative name to be assigned to the object or subpicture when it is saved. If svname is omitted, the actual name is used.

Description: This subroutine takes the object or subpicture name and saves it in the file or device assigned to SPUNCH. This subroutine allows specific parts of the main picture to be saved as opposed to calling IGDRON('SAVE') which will save the entire main picture. IGSAVE and IGDRON are the only subroutines that transmit hardware commands to output devices.

December 1980

IGSENS

Purpose: To obtain device-dependent information about a graphics device. This is an alternate entry name for AGSENS. See the AGSENS description.

IGSYM

Purpose: To add a character to the active subpicture/object, centered at the current position.

Calling Sequence:

FORTRAN: CALL IGSYM(char)

Parameters:

char is the character to be added. This will usually be a literal string.

Description: The character will be centered at the current position, usually for the purpose of marking a point on a graph. The character will be immune to any transformations that are applied directly to the subpicture. Thus, if the subpicture is transformed to fit within the screen boundaries, the character will not be stretched out of shape. The character will, however, be subject to any transformations applied to any higher-level picture.

Example:

```
CALL IGSYM('*')
```

IGTRAN

Purpose: To specify the transformation associated with a subpicture.

Calling Sequence:

FORTRAN: CALL IGTRAN(namsub[,type[,modif,...],...])

Parameters:

namsub is the name (four-character or internal) of the subpicture.

type is a four-character keyword specifying a type of basic transformation. The legal values are listed below.

modif are additional parameters describing the transformation. Each legal type must be followed by some fixed number of modif parameters.

Description: This subroutine specifies the transformation associated with the subpicture. IGTRAN may be called at any time, whether the subpicture is active or not. When IGDRON is called, the transformation is applied to the subpicture, to map it into the coordinate space of the next-higher-level picture. Then the transformation associated with the next-higher-level picture is applied, and so on. All transformations are applied before any viewport mappings specified by IGVWPT.

The transformation associated with the subpicture consists of a concatenated series of basic transformations. Each basic transformation is specified by a type keyword followed by some fixed number of modif parameters, as in the following list:

'MOVE',dx,dy	Translate by the displacement (dx,dy)
'MOV3',dx,dy,dz	Translate by the displacement (dx,dy,dz)
'ROTX',ang	Rotate ang radians clockwise around the X axis.
'ROTY',ang	Rotate ang radians clockwise around the Y axis.
'ROTZ',ang	Rotate ang radians clockwise around the Z axis.
'SCALE',s	Scale by the factor s.
'XSCALE',s	Scale x-coordinates by the factor s.

'YSCALE',s	Scale y-coordinates by the factor s.
'ZSCALE',s	Scale z-coordinates by the factor s.
'WINDOW',xl,xr,yb,yt	Scale and translate in X and Y such that the point (xl,yb) maps into (-1,-1) and the point (xr,yt) maps into (+1,+1).
'CURR'	Apply the current transformation; i.e., the transformation prior to the call to IGTRAN.
'PROJ',1./f,1./d	Project in perspective, with the viewpoint at (0,0,-d) and the maximum visible Z at f.

Internally, these basic transformations are represented by 4x4 matrices, which are multiplied together in the given order to obtain the transformation associated with the subpicture.

Examples:

```
CALL IGTRAN('PIC1','MOVE',.5,.5)
CALL IGTRAN('PIC2','WIND',0.,1023.,0.,1023.)
CALL IGTRAN('PIC3','ROTZ',PI,'SCALE',.1)
CALL IGTRAN('PIC4','ROTZ',DANGLE,'CURR')
```

IGTXT

Purpose: To add a transformable text string to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGTXT(string[,string,...])

Parameters:

string This may be one of the following:

(1) A string of up to 256 characters, terminated by the control operand <E>. Each character must have a printing ASCII equivalent. Control operands in the form <op> may be embedded in the string. The valid control operands are described below.

(2) A control operand changing the text scale or font. The valid control operands are described below.

Description: The text string is added to the active subpicture/object with the lower-left corner of the first character at the current position. Upon exit from IGTXT, the current position will be at the lower-right corner of the last character of the string.

If a transformation is applied to the subpicture, the text will be transformed along with the rest of the subpicture. The text itself may be scaled (see below) or printed in a different character set or font.

Whenever possible, the text will be generated by hardware. The default scale will be approximately .027 (depending on the kind of terminal). If the subpicture is scaled or rotated, or if the text is scaled or printed in a character set or font that is not available in hardware, the text will be generated in software.

Certain control operands may be embedded in the text string to perform local scaling and positioning operations. The valid control operands are:

<E> End of string.

<CRLF> Carriage return. The following text will be positioned one character space below the preceding text and at the X coordinate of the beginning of the string.

<BSUP> Begin superscript. The following text will be displaced upward by one-half the current character height and made one-half the current size.

<ESUP> End superscript. The following text will be displaced downward by the current character height and made twice the current size.

<BSUB> Begin subscript. The following text will be displaced downward by one-half the current character height and made one-half the current size.

<ESUB> End subscript. The following text will be displaced upward by the current character height and made twice the current size.

The scale of text may be locally specified by giving a sequence of two parameters. The first parameter is a control operand that specifies the kind of scaling operation and the second parameter specifies the scale. The valid sequences are:

'<ASCL>',scale Subsequent text will have the absolute size "scale", measured in subpicture coordinates.

'<RSCL>',scale Subsequent text will have a relative size that is "scale" times the current size.

The scale will revert to the default (normally the hardware character scale) if any IG subroutine other than IGFMT or IGTXT is called.

A character set or font may be specified by giving the sequence of parameters,

'<FONT>',cset

where cset is a keyword giving the name of the character set or font:

'STANDARD'	uppercase ASCII
'7ASCII'	full 7-bit ASCII
'SANSERIF.1'	
'SANSERIF.2'	
'SANSERIF.CART'	

```
'ROMAN.2A'
'ROMAN.2'
'ROMAN.3'
'ITALIC.2A'
'ITALIC.2'
'ITALIC.3'
'SCRIPT.1'
'SCRIPT.2'
'GREEK.1'
'GREEK.2A'
'GREEK.2'
'GREEK.CART'
'GREEK'
'GOTHIC.ENGLISH'
'GOTHIC.FRAKTUR'
'GOTHIC.ITALIAN'
'CYRILLIC.2'
```

These keywords must be spelled out in full. (This is an exception to the rule that, with most IG keywords, the first four letters are sufficient.) The character set or font will be reset to the default (normally uppercase ASCII) if any IG subroutine other than IGTXT, IGSYM, or IGFMT is called.

IGTXTH

Purpose: To add a hardware text string to the active subpicture/object.

Calling Sequence:

FORTRAN: CALL IGTXTH(string)

Parameters:

string This is a string of up to 256 characters, terminated by the control operand <E>. Each character must have a printing ASCII equivalent. Control operands in the form <op> may be embedded in the string. The valid control operands are described below.

Description: The text string is added to the active subpicture/object with the lower-left corner of the first character at the current position. Upon exit from IGTXTH, the current position will be at the lower-right corner of the last character of the string.

The text will be generated by hardware (except when the output device has no hardware character set, in which case the text will be generated in software, at a scale of .027). If a transformation is applied to the subpicture, the position of the text (i.e., the lower-left corner of the first character) will be transformed along with the rest of the subpicture. However, the text itself will not be scaled or rotated.

Certain control operands may be embedded in the text string to perform local scaling and positioning operations. Any scaled portion of the text will be generated in software. The valid control operands are:

<E> End of string.

<CRLF> Carriage return. The following text will be positioned one character space below the preceding text and at the X coordinate of the beginning of the string.

<BSUP> Begin superscript. The following text will be displaced upward by one-half the current character height and made one-half the current size.

<ESUP> End superscript. The following text will be displaced downward by the current character height and made twice the current size.

<BSUB> Begin subscript. The following text will be displaced downward by one-half the current character height and made one-half the current size.

<ESUB> End subscript. The following text will be displaced upward by the current character height and made twice the current size.

IGUSER

Purpose: To manipulate the value of the user word associated with a subpicture.

Calling Sequence:

FORTRAN: CALL IGUSER(namsub,putget,value)

Parameters:

namsub is the name (four-character or internal) of the subpicture.

putget is one of the following keywords:

'PUT ' Store value in the user word.
'GET ' Fetch value from the user word.

value is the value to be stored or fetched.

Description: Each subpicture has a single word (four bytes) reserved for the user. This is initially set to zero, but can be reset to a different value by calling this subroutine.

IGVEC

Purpose: To add multiple lines to the active subpicture/object.

Calling Sequences:

FORTRAN:
```
CALL IGVEC(nlines,xvec,yvec)
CALL IGVEC(nlines,intvec,xvec,yvec)
CALL IGVEC(nlines,intvec,movtyp,xvec,yvec)
CALL IGVEC(nlines,intvec,movtyp,nwords,
     xvec,yvec[,zvec])
```

Parameters:

nlines is the number of lines to add, i.e., the number of values to extract from the arrays xvec, yvec, and zvec.

intvec may be one of the following:

(1) A fullword-integer array of dimension nlines indicating move/draw for each line (0=move, 1=draw).
(2) A text string indicating a sequence of move/draw choices in a more compact way (see below).

If intvec is omitted, the first line is a move and all subsequent lines are draws.

movtyp may be one of the following:

(1) A fullword-integer array of dimension nlines, indicating relative/absolute for each line (0=relative, 1=absolute).
(2) A text string indicating a sequence of relative/absolute choices in a more compact way (see below).

If movtyp is omitted, all lines are absolute.

nwords is the number of fullwords separating values to be extracted from the arrays xvec, yvec, and zvec. If omitted, a value of 1 is assumed.

xvec,yvec,zvec

are REAL*4 arrays containing X, Y, and Z coordinates for the endpoint of each line. If

omitted, zvec is ignored, i.e., the Z coordinate is not changed.

Description: This subroutine performs the equivalent of a series of calls to IGMA, IGDA, IGMR, and IGDR, adding to the active subpicture/object a series of nlines lines. Each line may be a move (an invisible line) or a draw (a visible line). The arrays xvec, yvec, and zvec contain coordinates for the endpoint of each line. These coordinates may be absolute (as with IGMA or IGDA) or relative to the endpoint of the previous line (as with IGMR or IGDR).

The intvec parameter, when specified as a character string, forms an "expression" that generates a sequence of move/draw choices for the lines. This expression is interpreted character by character, as follows:

"M"	Current line is a move.
"D"	Current line is a draw.
" "	Go back to the beginning of the string.
")"	Go back to the last previous "(" and resume scanning.

For example, the string 'D ' represents an infinite series of draws and the string 'M(D)' represents a single move followed by an infinite series of draws.

The movtyp parameter, when specified as a character string, forms an "expression" that generates a sequence of relative/absolute choices for the lines. This expression is interpreted character by character, as follows:

"A"	Current line is absolute.
"R"	Current line is relative.
" "	Go back to the beginning of the string.
")"	Go back to the last previous "(" and resume scanning.

For example, the string 'A ' represents an infinite series of absolute lines and the string '(AR)' represents an infinite series which alternates between absolute and relative lines.

IGVWPT

Purpose: To specify the viewport associated with a subpicture.

Calling Sequence:

FORTRAN: CALL IGVWPT(namsub,xleft,xright,ybotom,ytop)

Parameters:

namsub is the name (four-character or internal) of the subpicture.

xleft is the X coordinate of the left edge of the viewport.

xright is the X coordinate of the right edge of the viewport.

ybotom is the Y coordinate of the bottom edge of the viewport.

ytop is the Y coordinate of the top edge of the viewport.

Description: This subroutine specifies, in -1 to +1 screen coordinates, the viewport into which the subpicture will be mapped. IGVWPT may be called at any time, whether the subpicture is active or not. This viewport mapping will be applied after all transformations specified by IGTRAN.

Under the viewport mapping, the point (-1,-1) is mapped into (xleft,ybotom) and the point (+1,+1) is mapped into the point (xright,ytop). The subpicture is then clipped at the coordinates xleft, xright, ybotom, ytop as well as at the edges of the screen. Note that if xright-xleft does not equal ytop-ybotom, this mapping will stretch the subpicture by a different amount in the X direction than in the Y direction.

Examples: This mapping can be combined with the 'WIND' type of IGTRAN transformation to form the traditional window-to-viewport transformation:

```
CALL IGTRAN('PICT','WIND',0.,100.,50.,150.)
CALL IGVWPT('PICT',-.9,.0,.0,.9)
```

IGXYIN

Purpose: To perform graphic coordinate input, by reading the locator coordinates.

Calling Sequence:

FORTRAN: icode = IGXYIN(xin,yin)

Values Returned:

icode is an integer code for the keyboard key that was pressed to return the locator coordinates.

xin,yin are the locator coordinates, expressed in the coordinate space of the active subpicture.

Description: This subroutine implicitly calls IGDRON('TERMINAL') to update the screen image. It then waits for the user to position the locator. (This may be a tracking cross, cursor, pair of crosshairs, or some other mechanism. The details vary according to terminal type. See Appendix I.) When the user presses any keyboard key, the subroutine reads the locator coordinates and expresses them in the coordinate space of the active subpicture. The subroutine also returns an integer code icode that represents the keyboard key that was pressed (see Table 1 in the section "Graphic Input").

The coordinates xin and yin can be used to draw the active subpicture interactively. If these coordinates are passed directly to IGDA, a line will be drawn from the current position to the locator position.

Under the following circumstances, the operation of IGXYIN is ambiguous: (1) If the active subpicture is a subpicture of an object, and if this object has several instances, then the active subpicture has several different transformations. It follows that the locator coordinates are undefined relative to the coordinate space of the active subpicture. (2) If an object is active, then there is no active subpicture and the locator coordinates are undefined. In this case, a call to IGXYIN will result in an error.

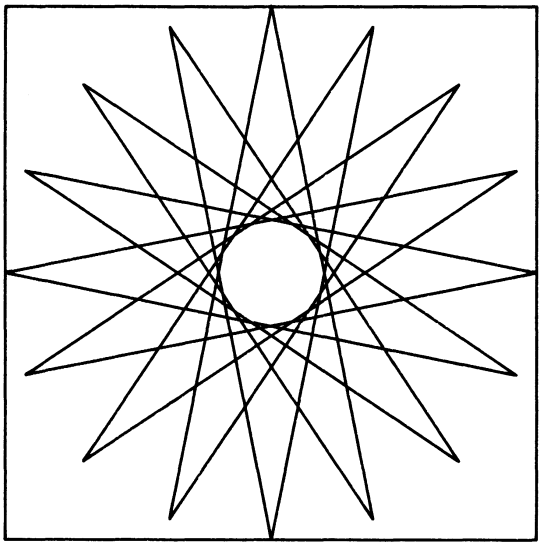
APPENDIX C: EXAMPLE PROGRAMS

This appendix contains example programs which illustrate some common graphics applications.

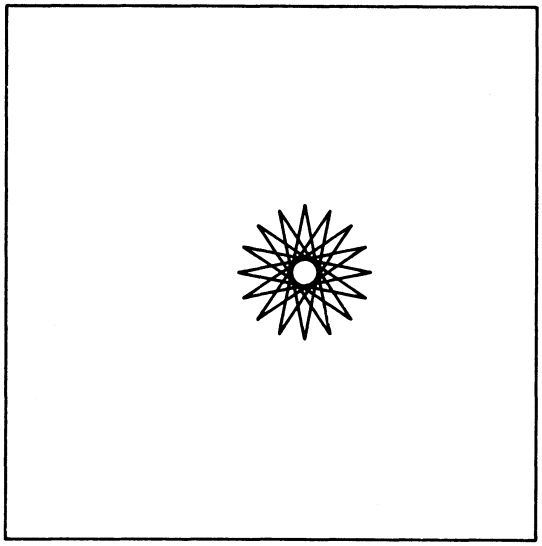
The source for these examples resides in the file GRAF:IG.EXAMP.S. Example 1 starts at line 1000, Example 2 at line 2000, etc.

```
C===============================================================C
C                     IG EXAMPLE PROGRAM 1                      C
C                                                               C
C           ILLUSTRATES BASIC PICTURE MANIPULATIONS             C
C===============================================================C
C---------------------------------------------------------------C
C                     INITIALIZE THE SYSTEM                     C
C---------------------------------------------------------------C
      CALL IGINIT
C---------------------------------------------------------------C
C               BEGIN DESCRIPTION OF PICTURE 'PLOT'             C
C---------------------------------------------------------------C
      CALL IGBGNS('PLOT')
C---------------------------------------------------------------C
C                   ADD LINES TO THE PICTURE                    C
C---------------------------------------------------------------C
      CALL IGMA(0.0,1.0)
      DO 10 I=1,16
       A=.875*3.14159*I
 10    CALL IGDA(SIN(A),COS(A))
C---------------------------------------------------------------C
C                TERMINATE DESCRIPTION OF 'PLOT'                C
C---------------------------------------------------------------C
      CALL IGENDS('PLOT')
C---------------------------------------------------------------C
C              DISPLAY THE PICTURE ON THE TERMINAL              C
C---------------------------------------------------------------C
 1    CALL IGDRON('TERMINAL')
C---------------------------------------------------------------C
C                PLOT ON CALCOMP IF USER DESIRES                C
C---------------------------------------------------------------C
      CALL ASKPLT
C---------------------------------------------------------------C
C   REDISPLAY THE PICTURE SHIFTED TO THE LEFT AND 1/4 AS BIG    C
C---------------------------------------------------------------C
      CALL IGTRAN('PLOT','MOVE',.5,.0,'SCALE',.25)
 2    CALL IGDRON('TERMINAL')
      CALL ASKPLT
C---------------------------------------------------------------C
C                RESPECIFY DEFINITION OF 'PLOT'                 C
C---------------------------------------------------------------C
      CALL IGBGNS('PLOT')
      DO 20 I=1,201
        A=((I-1)/200.) * 6.2830
        Y=SIN(A)*SIN(A*10.)
 20     CALL IGDA(A,Y)
      CALL IGENDS('PLOT')
C
 3    CALL IGDRON('TERMINAL')
      CALL ASKPLT
```

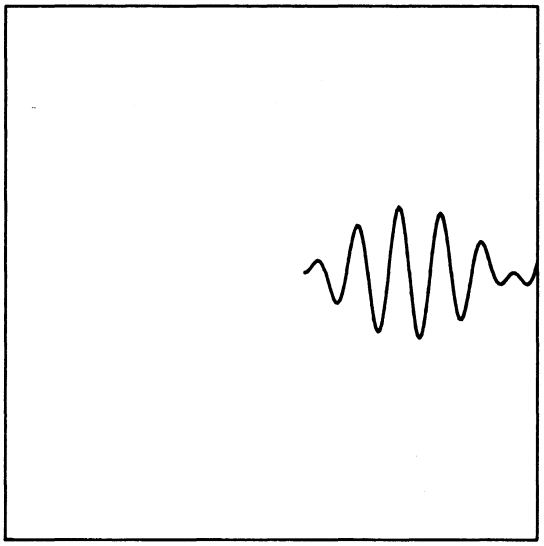
```
C-------------------------------------------------------------C
C            REDISPLAY THE PICTURE CENTERED AND SHRUNK         C
C-------------------------------------------------------------C
      CALL IGTRAN('PLOT','MOVE',-3.14, 0.00,'SCALE',1./3.14)
 4    CALL IGDRON('TERMINAL')
      CALL ASKPLT
      END
C=============================================================C
C               ASK USER IF HE/SHE WANTS HARD COPY            C
C=============================================================C
      SUBROUTINE ASKPLT
      INTEGER YES/'Y'/
C
      PRINT 999
 999  FORMAT('&PLOT? ')
      READ(5,998) IANSWR
 998  FORMAT(A1)
C
      IF (IANSWR .EQ. YES) CALL IGDRON('CALCOMP')
      RETURN
      END
```



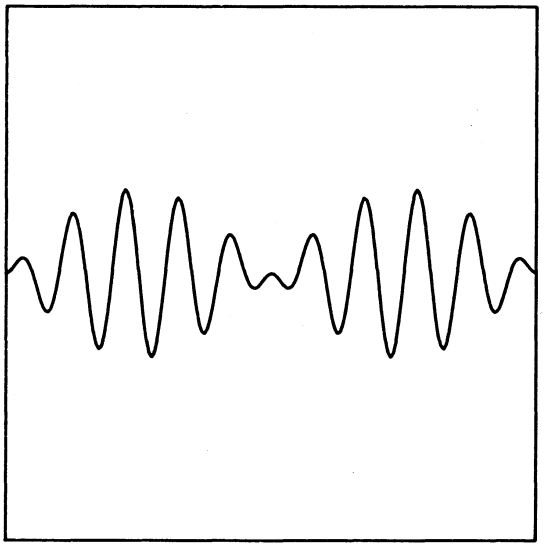
Screen After Statement 1



Screen After Statement 2



Screen After Statement 3



Screen After Statement 4

```
C=============================================================C
C                     IG EXAMPLE PROGRAM 2                    C
C                                                             C
C     ILLUSTRATING A SIMPLE SET OF GRAPH PLOTTING ROUTINES    C
C=============================================================C
      DIMENSION X(101),Y(101)
      CALL IGINIT
C-------------------------------------------------------------C
C                 GENERATE DATA TO BE PLOTTED                 C
C-------------------------------------------------------------C
      DO 10 I=1,101
        XX = (I-1)*.0628
        X(I) = XX
 10     Y(I) = SIN(20.*XX)+SIN(22.*XX)
C-------------------------------------------------------------C
C                         PLOT THE DATA                       C
C-------------------------------------------------------------C
 1    CALL PLOT(101,X,Y)
      PAUSE 'RESTART FOR NEXT PLOT'
C-------------------------------------------------------------C
C         GENERATE NEW DATA TO BE PLOTTED AT SAME SCALE       C
C-------------------------------------------------------------C
      DO 20 I=1,101
 20     Y(I) = SIN(20.*X(I)) + SIN(21.*X(I))
C-------------------------------------------------------------C
C           PLOT THE NEW DATA WITH SAME AXIS SCALING          C
C-------------------------------------------------------------C
      CALL LINE(101,X,Y)
 2    CALL IGDRON('TERM')
      PAUSE 'RESTART TO BLOW UP PLOT'
C-------------------------------------------------------------C
C                ZOOM IN ON A REGION OF THE PLOT              C
C-------------------------------------------------------------C
      CALL IGCTNS('LINE')
      PRINT 30
 30   FORMAT(' ENTER LOWER LEFT CORNER')
      CALL IGXYIN(XMIN,YMIN)
      PRINT 40
 40   FORMAT(' ENTER UPPER RIGHT CORNER')
      CALL IGXYIN(XMAX,YMAX)
      CALL AXES(XMIN,XMAX,YMIN,YMAX)
 3    CALL IGDRON('TERMINAL')
      STOP
      END
```

```
C=============================================================C
C             PLOT A GRAPH OF TWO VECTORS OF DATA             C
C                                                             C
      SUBROUTINE PLOT(N,XVEC,YVEC)
C                                                             C
C N   = NUMBER OF LINES TO PLOT                               C
C XVEC= VECTOR OF X VALUES                                    C
C YVEC= VECTOR OF Y VALUES                                    C
C=============================================================C
      DIMENSION XVEC(1),YVEC(1)
C-------------------------------------------------------------C
C               FIND MIN,MAX IN ARRAYS XVEC,YVEC              C
C-------------------------------------------------------------C
      XMIN=XVEC(1)
      XMAX=XVEC(1)
      YMIN=YVEC(1)
      YMAX=YVEC(1)
        DO 10 I=2,N
        IF(XVEC(I).GT.XMAX) XMAX=XVEC(I)
        IF(XVEC(I).LT.XMIN) XMIN=XVEC(I)
        IF(YVEC(I).GT.YMAX) YMAX=YVEC(I)
        IF(YVEC(I).LT.YMIN) YMIN=YVEC(I)
 10     CONTINUE
C-------------------------------------------------------------C
C                          PLOT THE DATA                      C
C-------------------------------------------------------------C
      CALL LINE(N,XVEC,YVEC)
C-------------------------------------------------------------C
C               DRAW THE AXES AND SCALE THE DATA              C
C-------------------------------------------------------------C
      CALL AXES(XMIN,XMAX,YMIN,YMAX)
      CALL IGDRON('TERMINAL')
      RETURN
      END
```

```
C=============================================================C
C               DRAW THE AXES AND SCALE THE DATA              C
C                                                             C
      SUBROUTINE AXES(XMIN,XMAX,YMIN,YMAX)
C                                                             C
C XMIN = MINIMUM X VALUE                                      C
C XMAX = MAXIMUM X VALUE                                      C
C YMIN = MINIMUM Y VALUE                                      C
C YMAX = MAXIMUM Y VALUE                                      C
C=============================================================C
      CALL IGBGNS('AXES')
C-------------------------------------------------------------C
C                   DRAW AND LABEL THE Y AXIS                 C
C-------------------------------------------------------------C
       CALL IGMA(-.90,+.55)
       CALL IGFMT(YMAX,'F',5,2)
C
       CALL IGMA(-.70,+.60)
       CALL IGDA(-.70,-.60)
C
       CALL IGMA(-.90,-.60)
       CALL IGFMT(YMIN,'F',5,2)
C-------------------------------------------------------------C
C                   DRAW AND LABEL THE X AXIS                 C
C-------------------------------------------------------------C
       CALL IGMA(-.75,-.75)
       CALL IGFMT(XMIN,'F',5,2)
C
       CALL IGMA(-.70,-.60)
       CALL IGDA(+.70,-.60)
C
       CALL IGMA(+.65,-.75)
       CALL IGFMT(XMAX,'F',5,2)
      CALL IGENDS('AXES')
C-------------------------------------------------------------C
C                        SCALE THE DATA                       C
C-------------------------------------------------------------C
      CALL IGTRAN('LINE','WINDOW',XMIN,XMAX,YMIN,YMAX)
      CALL IGVWPT('LINE',-.70,+.70,-.60,+.60)
      RETURN
      END
```

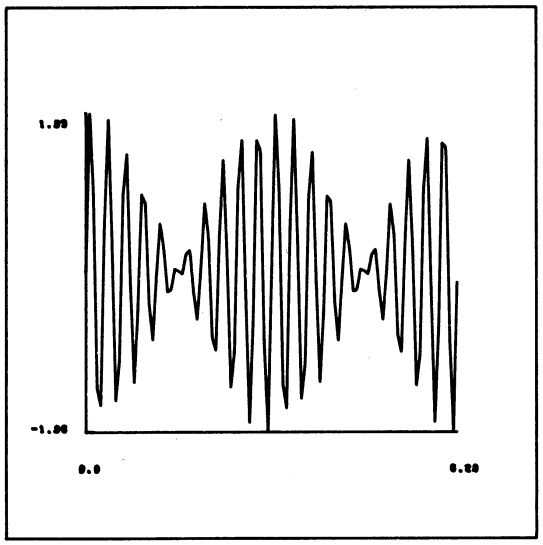
```
C=============================================================C
C                          PLOT THE DATA                      C
C                                                             C
      SUBROUTINE LINE(N,XVEC,YVEC)
C                                                             C
C N    = NUMBER OF LINES IN VECTOR                            C
C XVEC = VECTOR OF X VALUES                                   C
C YVEC = VECTOR OF Y VALUES                                   C
C=============================================================C
      DIMENSION XVEC(1),YVEC(1)
      CALL IGBGNS('LINE')
       CALL IGVEC(N,XVEC,YVEC)
      CALL IGENDS('LINE')
      RETURN
      END
```

Notes on Example 2:

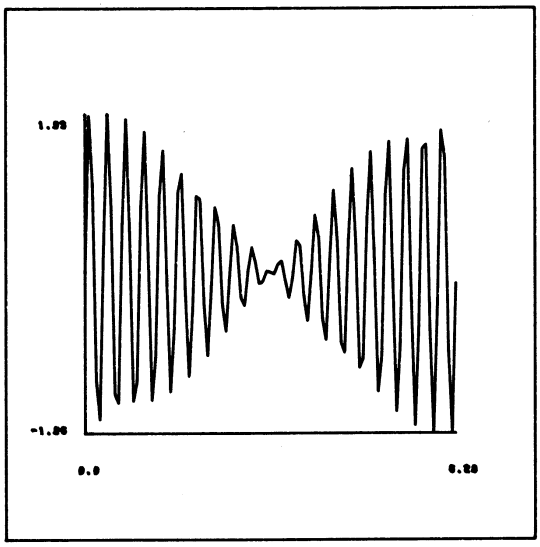
(1) The 'AXES' picture is not transformed. Thus, its coordinate space is the same as the screen coordinate space. The 'LINE' picture contains the plotted data. The AXES subroutine transforms 'LINE' by placing a window around the subregion to be plotted and mapping this window into a viewport which coincides with the rectangular region enclosed by the axis lines.

(2) In the main program, the call to IGCTNS('LINE') makes 'LINE' the active picture so that the following calls to IGXYIN will return values in the coordinate space of 'LINE', i.e., the coordinate space of the plotted data.

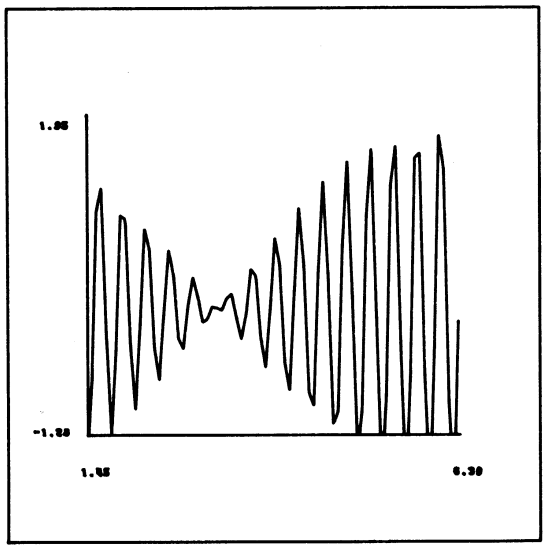
(3) The window on 'LINE' is modified within the AXES subroutine. This allows different subregions of 'LINE' to be displayed without the need to alter the contents of 'LINE'.



Screen After Statement 1



Screen After Statement 2



Screen After Statement 3

```
C=============================================================C
C                    IG EXAMPLE PROGRAM 3                     C
C                                                             C
C         ILLUSTRATES USE OF IGXYIN AND PICTURE BLOW-UP       C
C=============================================================C
      INTEGER*2 LEN2
      INTEGER*4 BUFFER(20)
      CALL IGINIT
      YTOP=AGSENS('TERMINAL','YSIZE')
      CALL IGBGNS('PICT')
      PRINT 9
 9    FORMAT(' KEYS: 0=MOVE'   /
     .       '       1=DRAW'   /
     .       '       T=TEXT'   /
     .       '       B=BLOW UP'/
     .       '       R=RESTORE'/
     .       '       P=PLOT'   /
     .       '       S=STOP'   )
C-------------------------------------------------------------C
C             READ LOCATOR COORDINATES AND KEY CODE           C
C-------------------------------------------------------------C
 1    ICODE = IGXYIN(XIN,YIN)
      IF (ICODE .EQ. 0) CALL IGMA(XIN,YIN)
      IF (ICODE .EQ. 1) CALL IGDA(XIN,YIN)
      IF (ICODE .EQ. 36) GOTO 100
      IF (ICODE .EQ. 18) GOTO 200
      IF (ICODE .EQ. 34) GOTO 300
      IF (ICODE .EQ. 32) CALL IGDRON('CALCOMP')
      IF (ICODE .EQ. 35) STOP
      GOTO 1
C-------------------------------------------------------------C
C                  ADD TEXT STRING TO PICTURE                 C
C-------------------------------------------------------------C
 100  CALL IGMA(XIN,YIN)
      PRINT 199
 199  FORMAT('&ENTER TEXT:')
      CALL SCARDS(BUFFER,LEN2,0,LINEN)
      LEN4=LEN2
      CALL IGFMT(BUFFER,'A',LEN4)
      GOTO 1
C-------------------------------------------------------------C
C             BLOW-UP RECTANGULAR REGION OF PICTURE           C
C-------------------------------------------------------------C
 200  PRINT 299
 299  FORMAT(' ENTER LOWER LEFT CORNER')
      CALL IGXYIN(XLEFT,YLOWER)
      PRINT 298
 298  FORMAT(' ENTER UPPER RIGHT CORNER')
      CALL IGXYIN(XRIGHT,YUPPER)
      CALL IGTRAN('PICT','WINDOW',XLEFT,XRIGHT,YLOWER,YUPPER)
      ASPECT = (YUPPER-YLOWER)/(XRIGHT-XLEFT)
```

```
      IF (ASPECT .LT. YTOP)
     .   CALL IGVWPT('PICT',-1.,+1.,-ASPECT, ASPECT)
      IF (ASPECT .GE. YTOP)
     .   CALL IGVWPT('PICT',-YTOP/ASPECT,YTOP/ASPECT,-YTOP,YTOP)
      GOTO 1
C-------------------------------------------------------------C
C                  RESTORE TO ORIGINAL SIZE                   C
C-------------------------------------------------------------C
 300  CALL IGTRAN('PICT','WINDOW',-1.,+1.,-1.,+1.)
      CALL IGVWPT('PICT',         -1.,+1.,-1.,+1.)
      GOTO 1
C
      END
```

Notes on Example 3:

(1) The program never has to call IGDRON('TERMINAL') since this is implicitly done by the IGXYIN subroutine.

(2) 'PICT' is always the active picture. The locator coordinates are always returned in the coordinate space of 'PICT' even if it is blown up.

(3) To blow up the picture, the user chooses a window via IGXYIN calls. The program uses the following criteria to choose a viewport into which the window is to be mapped:

   (a) The viewport must have the same aspect ratio (height/width) as the window so that the picture is not distorted.

   (b) The viewport's maximum X extent must fit within -1 to +1 and its maximum Y extent must fit within -YTOP to +YTOP.

(4) Text strings may be read in and added to the picture via IGFMT. The length of each string is passed to IGFMT as a parameter.

```
C=============================================================C
C                     IG EXAMPLE PROGRAM 4                    C
C                                                             C
C              ILLUSTRATING USE OF PICK OPERATIONS            C
C=============================================================C
      CALL IGINIT
      CALL IGCTRL('TERMINAL','KEEP',1)
C-------------------------------------------------------------C
C      CREATE A PICTURE 'PICT' AND A BUNCH OF SUBPICTURES     C
C-------------------------------------------------------------C
      CALL IGBGNS('PICT')
      DO 20 I=3,6
      NAME=IGBGNS(0,'SCALE',.2,'MOVE',(I-4.5)/2.,0.0)
      CALL IGMA(0.,1.)
      DO 10 J=1,I
      A=J*6.28/I
 10   CALL IGDA(SIN(A),COS(A))
 20   CALL IGENDS(NAME)
      CALL IGENDS('PICT')
C-------------------------------------------------------------C
C               CREATE THE GLOBAL COMMAND MENU                C
C-------------------------------------------------------------C
 30   CALL IGBGNS('MENU')
      CALL MENU('TRANSFORM<E>',.3)
      CALL MENU('DELETE<E>'   ,.0)
      CALL MENU('STOP<E>'    ,-.3)
      CALL IGENDS('MENU')
C-------------------------------------------------------------C
C                     PICK A GLOBAL COMMAND                   C
C-------------------------------------------------------------C
 40   CALL IGCTRL('TERMINAL','ERASE')
 1    JUMP=IGPIKS('TRAN','DELE','STOP')
      GOTO (1000,2000,3000),JUMP
C-------------------------------------------------------------C
C                                                             C
C               TRANSFORM A SUBPICTURE OF 'PICT'              C
C                                                             C
C-------------------------------------------------------------C
 1000 PRINT 1099
 1099 FORMAT(' PICK ITEM TO TRANSFORM')
      CALL IGPIKS('PICT')
      NAMTRN=IGPIKN(1)
C-------------------------------------------------------------C
C               SWITCH TO TRANSFORMATION MENU                 C
C-------------------------------------------------------------C
      CALL IGBGNS('MENU')
       CALL MENU('MOVE<E>',   .45)
       CALL MENU('SCALE<E>',  .15)
       CALL MENU('ROTATE<E>',-.15)
       CALL MENU('DONE<E>',  -.45)
      CALL IGENDS('MENU')
```

```
C-------------------------------------------------------------C
C                   PICK A TRANSFORMATION                     C
C-------------------------------------------------------------C
 1001 JUMP=IGPIKS('MOVE','SCAL','ROTA','DONE')
      GOTO (1100,1200,1300,30),JUMP
C-------------------------------------------------------------C
C                            MOVE                             C
C-------------------------------------------------------------C
 1100 PRINT 1199
 1199 FORMAT(' POINT TO CURRENT LOCATION')
      CALL IGXYIN(X1,Y1)
      PRINT 1198
 1198 FORMAT(' POINT TO NEW LOCATION')
      CALL IGXYIN(X2,Y2)
      CALL IGTRAN(NAMTRN,'CURRENT','MOVE',X2-X1,Y2-Y1)
      GOTO 1001
C-------------------------------------------------------------C
C                            SCALE                            C
C-------------------------------------------------------------C
 1200 PRINT 1299
 1299 FORMAT('&ENTER SCALE FACTOR:')
      READ 1999,FACTOR
      CALL IGTRAN(NAMTRN,'SCALE',FACTOR,'CURRENT')
      GOTO 1001
C-------------------------------------------------------------C
C                           ROTATE                            C
C-------------------------------------------------------------C
 1300 PRINT 1399
 1399 FORMAT('&ENTER ANGLE:')
      READ 1999,ANGLE
      CALL IGTRAN(NAMTRN,'ROTZ',ANGLE,'CURRENT')
      GOTO 1001
C
 1999 FORMAT(F10.0)
C-------------------------------------------------------------C
C                                                             C
C               DELETE A SUBPICTURE OF 'PICT'                 C
C                                                             C
C-------------------------------------------------------------C
 2000 PRINT 2099
 2099 FORMAT(' PICK ITEM TO DELETE')
      CALL IGPIKS('PICT')
      CALL IGDELS(IGPIKN(1))
      GOTO 40
C-------------------------------------------------------------C
C                                                             C
C                            STOP                             C
C                                                             C
C-------------------------------------------------------------C
 3000 STOP
      END
```

```
C===============================================================C
C                   PLACE AN ITEM IN THE MENU                   C
C                                                               C
      SUBROUTINE MENU(STRING,Y)
C                                                               C
C STRING = TEXT STRING TERMINATED BY "<E>"                      C
C          STRING BECOMES A MENU ITEM (SUBPICTURE)              C
C          FIRST FOUR CHARS BECOME NAME OF MENU ITEM            C
C Y      = Y COORDINATE OF MENU ITEM                            C
C===============================================================C
      DIMENSION STRING(1)
      CALL IGBGNS(STRING)
      CALL IGMA(-1.,Y)
      CALL IGTXT(STRING)
      CALL IGENDS(STRING)
      RETURN
      END
```

Notes on Example 4:

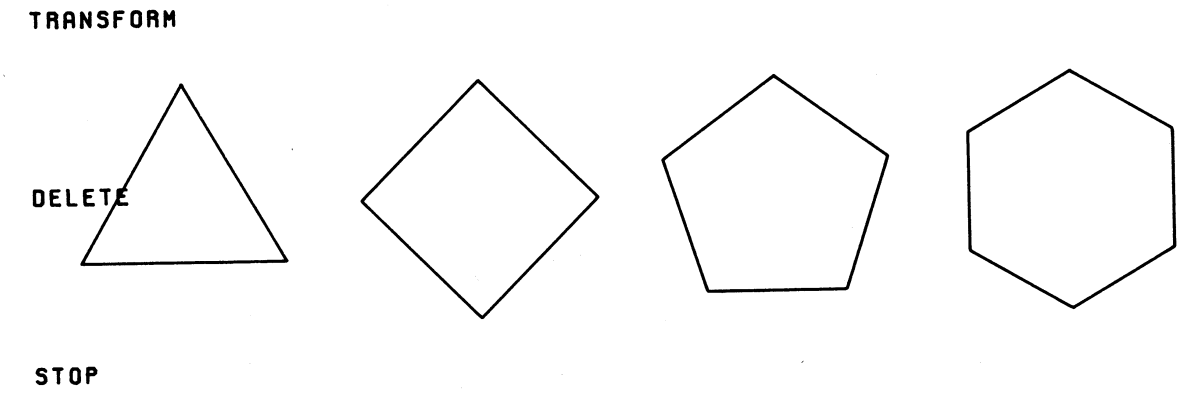
(1) Each menu item is a separate subpicture, consisting of a text string. A call to IGPIKS with a list of these subpictures allows the user to pick one subpicture from the list. Hits from any other part of the screen are ignored.

(2) The 'MENU' picture contains all menu item subpictures. When 'MENU' is respecified, the old menu is deleted and a new one is created.

(3) A call to IGPIKS('PICT') picks one of the four subpictures in 'PICT' since there are no lines in the main body of 'PICT'. The name of this subpicture is returned as IGPIKN(1).

(4) Several transformations may be picked and applied to this subpicture. Scalings and rotations are concatenated before the current transformation, while translations are concatenated after the current transformation. In this way, scalings and rotations are always applied before translations.

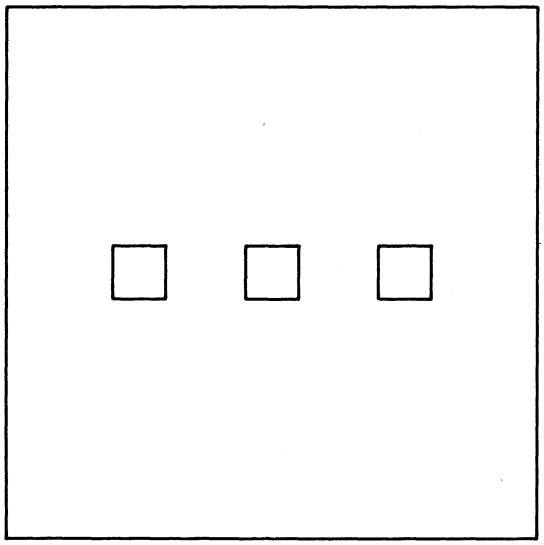
(5) If the terminal is a storage-tube-type device, each scaling, rotation or translation normally requires erasing the screen and regenerating the entire screen image. To avoid this, the program calls IGCTRL to enable keep mode. Subsequently, the program calls IGCTRL to erase the screen before displaying the global command menu. This gets rid of any junk images left behind during the preceding round of transformations. These calls to IGCTRL are ignored if the terminal is not a storage-tube-type device.



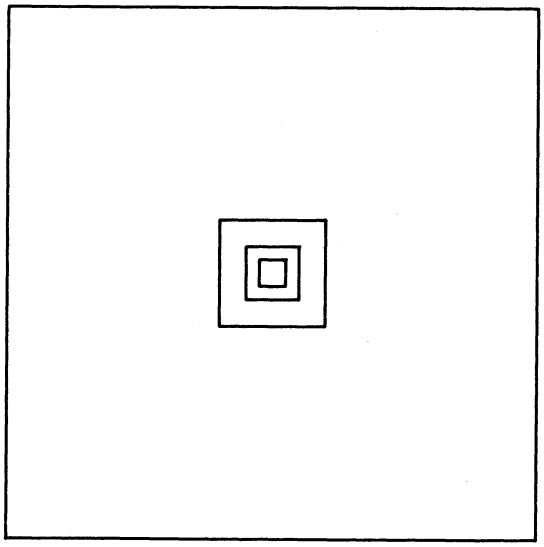
Screen After Statement 1

```
C===============================================================C
C                     IG EXAMPLE PROGRAM 5                      C
C                                                               C
C                  ILLUSTRATES USE OF OBJECTS                   C
C===============================================================C
      CALL IGINIT
      REAL*4 XSQR(5)/+.1,-.1,-.1,+.1,+.1/,
     .       YSQR(5)/+.1,+.1,-.1,-.1,+.1/
      REAL*4 XTRI(4)/+.0,-.1,+.1,+.0/,
     .       YTRI(4)/+.1,-.1,-.1,+.1/
C
      CALL IGBGNO('THING')
      CALL IGVEC(5,XSQR,YSQR)
      CALL IGENDO('THING')
C
      INST1=IGPUTO('THING')
      INST2=IGPUTO('THING',0     ,'MOVE',-.5,+.0)
      CALL  IGPUTO('THING','INS3','MOVE',+.5,+.0)
 1    CALL IGDRON('TERMINAL')
      PAUSE
C---------------------------------------------------------------C
C                TRANSFORM TWO OF THE INSTANCES                 C
C---------------------------------------------------------------C
      CALL IGTRAN(INST2,'SCALE',2.0)
      CALL IGTRAN('INS3','SCALE',0.5)
 2    CALL IGDRON('TERMINAL')
      PAUSE
C---------------------------------------------------------------C
C             RESPECIFY THE DEFINITION OF THE OBJECT            C
C---------------------------------------------------------------C
      CALL IGBGNO('THING')
      CALL IGVEC(4,XTRI,YTRI)
      CALL IGENDO('THING')
 3    CALL IGDRON('TERMINAL')
      PAUSE
C---------------------------------------------------------------C
C         DEFINE ANOTHER OBJECT AND ATTACH IT TO 'INS3'         C
C                        DELETE INST2                           C
C---------------------------------------------------------------C
      CALL IGBGNO('SQAR')
      CALL IGVEC(5,XSQR,YSQR)
      CALL IGENDO('SQAR')
      CALL IGPUTO('SQAR','INS3')
C
      CALL IGDELS(INST2)
 4    CALL IGDRON('TERMINAL')
      PAUSE
```

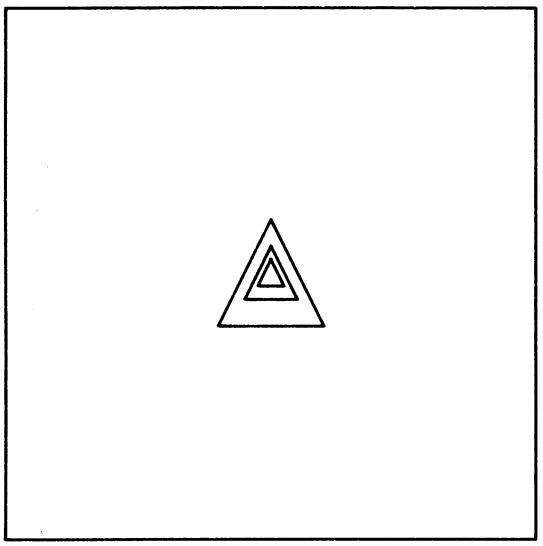
```
C---------------------------------------------------------------C
C               DELETE THE DEFINITION OF 'THING',               C
C       AUTOMATICALLY DELETING ALL INSTANCES OF 'THING'         C
C---------------------------------------------------------------C
      CALL IGDELO('THING')
 5    CALL IGDRON('TERMINAL')
      STOP
      END
```



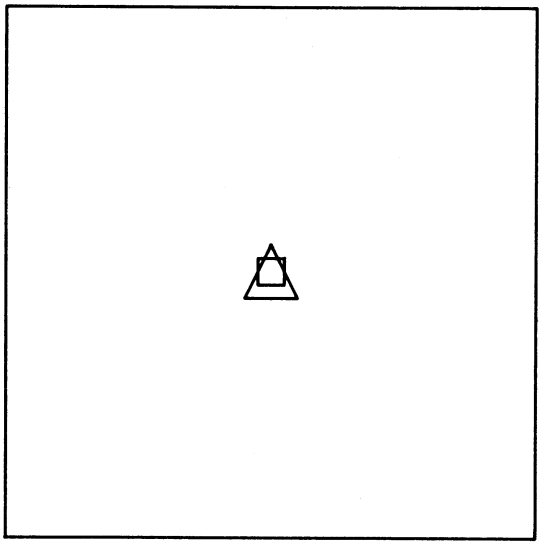
Screen After Statement 1



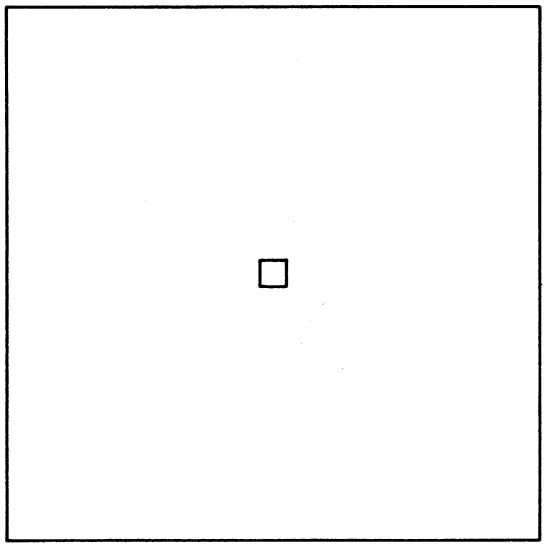
Screen After Statement 2



Screen After Statement 3



Screen After Statement 4



Screen After Statement 5

APPENDIX D: ERROR MESSAGES

Any errors detected in parameters to IG subroutines result in an internal call to the IG error monitor. This prints a three-line error message giving the nature of the error, the location of the call that produced the error, and the action to be taken if the user attempts to continue execution via the $RESTART command. The error monitor then calls MTS.

NATURE OF THE ERROR

The messages giving the nature of the error are listed below, in prototype form. Several of these refer to subroutines that are either obsolete or currently undocumented in this volume. The notation nnnnnn denotes the name of a subpicture or object, which may be either a six-digit hexadecimal number (an IG-created, internal name) or four characters enclosed in primes (a user-created, external name).

nnnnnn already exists and will be altered

> Upon a call to IGLOAD with 'PRESERVE' in effect, one of the reloaded subpictures had the same name as the existing subpicture nnnnnn. The contents of nnnnnn will be replaced by the contents of the reloaded subpicture.

Attempt to delete nonexistent object

> The parameter to IGDELO did not contain the name of an existing object.

Attempt to put data into bud block

> A call was made to one of the picture-description subroutines (IGDA, IGVEC, IGTXT, IGATTB, etc.) while the active subpicture was one that was defined using the EGP option.

Attempt to reference nonexistent subpicture "nnnnnn"

> A parameter that should have contained the name of an existing subpicture contained nnnnnn instead.

Auxiliary device "dddddd" already exists

> The first parameter to IGAUXD contained a device keyword dddddd that was already in use.

Bad (negative) value in INTVEC

The intvec parameter to IGVEC was not a fullword-integer array containing nonnegative values.

Bad (negative) value in MOVTYP

The movtyp parameter to IGVEC was not a fullword-integer array containing nonnegative values.

Bad character in INTVEC

The intvec parameter to IGVEC contained a character other than one of "MD() ".

Bad character in MOVTYP

The movtyp parameter to IGVEC contained a character other than one of "AR() ".

Bad format code "f"

The format parameter to IGFMT or IGFMTH was f instead of one of the legal values. See the descriptions of IGFMT and IGFMTH in Appendix B.

Bad length specified in IGTEXT call

The second parameter to the (obsolete) subroutine IGTEXT was not an integer in the range 1≤n≤256.

Bad nibble buffer length in IGLOAD

In reloading a set of objects, IGLOAD encountered bad data.

Call to IGENDO before IGENDS-ing all its subpictures

IGENDO was called while a subpicture was still active.

CALL IGENDS(mmmmmm) when active subpicture was "nnnnnn"

The parameter to IGENDS contained mmmmmm instead of the name nnnnnn of the active subpicture.

Direction vector must be nonzero

In a call to IGATTS or IGATTB, the 'TPLANE' attribute keyword was followed by a zero vector.

EGP object is illegal for this IGINFO call

A call to IGINFO asked for 'VWPT', 'CURRENT', 'MINIMUM', or 'MAXIMUM' for a subpicture that was defined using the EGP option.

Illegal character set/font with index "nnnn" encountered

The internal character set/font index used by IGDRON had the illegal value nnnn. This error was probably caused by some earlier error that destroyed part of the internal data structure.

Illegal input ordering "rrrrrr"-"ssssss"

In reloading a set of objects, IGLOAD encountered records that were out of sequence. This error may have been caused by the object-saving file not being emptied before the objects were saved.

Illegal name given in IGRNAM call

Either (1) the first parameter to IGRNAM was not the name of an existing subpicture, or (2) the second parameter to IGRNAM was not a four-character name beginning with a letter.

Illegal opcode in call by EGP in BLOOMBUD

An EGP called IG with a bad opcode.

Invalid device type "dddddd"

The device parameter to AGSENS, DGSENS, IGSENS, IGCTRL, or IGDRON was dddddd instead of 'TERM', 'CALC', 'SAVE', or a user-defined keyword for an auxiliary device.

Invalid file given for "dddddd";
$RESTART will cause default to plotter

The ddrfil parameter to IGAUXD was not one of the legal DDR file names. See the description of IGAUXD in Appendix B. Upon a $RESTART, the CalComp DDR '*IG.CCMP ' will be used instead.

Invalid parameter value in call to IGFMT

In a call to IGFMT, one of the parameters nchars or ndec did not have a legal value.

Invalid subpicture name given to IGCVTC

Either the frmsub or the tosub parameter to IGCVTC was not the name of an existing subpicture.

Invalid IGATTB parameter "pppppp"

One of the attrib keyword parameters to IGATTB either was invalid or was not followed by the right number of legal atval parameters.

Invalid IGATTS parameter "pppppp"

One of the attrib keyword parameters to IGATTS either was invalid or was not followed by a legal atval parameter.

Invalid IGINFO parameter "pppppp"

One of the inftyp keyword parameters to IGINFO either was invalid or was not followed by the right number of legal val parameters.

Invalid IGLIKE parameter "pppppp"

One of the attrib keyword parameters to IGLIKE was invalid.

IGLOAD unable to read file or device "fdname"

Either fdname does not exist or the user has no access to this file or device.

IGRNAM call specifies already existing name "nnnnnn" as new name

The second parameter to IGRNAM contained the name nnnnnn of an existing subpicture or object.

Nonexistent font "ffffffff"; the standard font will be used

The cset parameter to IGATTS or IGTXT was ffffffff instead of one of the legal values. For a list of the legal character sets/fonts, see Appendix E.

NWORDS must be positive

The nwords parameter to IGVEC was not a positive integer.

Object "nnnnnn" does not exist

A parameter that should have contained the name of an existing object contained nnnnnn instead.

Object name used as a subpicture name

A parameter that should have contained the name of a subpicture contained the name of an object instead.

Parameter given to IGINFO was not a subpicture

A call to IGINFO asked for 'VWPT', 'CURRENT', 'MINIMUM', or 'MAXIMUM' when the first parameter was not the name of a subpicture.

Parameter to IGPIKS or IGPIKC not a subpicture

One of the list of parameters to IGPIKS or IGPIKC did not contain the name of an existing subpicture.

Record not a begin-object record at line nnnn.nnn

In reloading a set of objects, IGLOAD encountered some other kind of record where there should have been a begin-object record. This error may have been caused by the object-saving file not being emptied before the objects were saved.

String terminated by nonprinting character

The string parameter to IGTXT or IGTXTH contained a character that did not have a printing ASCII equivalent. The string up to but not including the bad character will be inserted. This error may have been caused by the string not being terminated with the control operand <E>.

Subpicture name nnnnnn used as an object name

A parameter that should have contained the name of an object contained the name nnnnnn of a subpicture instead.

Too few parameters following "<cccccc>"

One of the IGTXT control operands '<ASCL>', '<RSCL>', or '<FONT>' was not followed by the right number of parameters.

Too many calls to IGENDS

More calls were made to IGENDS than were made to IGBGNS and IGCTNS combined.

Truncated record encountered by IGLOAD at line nnnn.nnn

In reloading a set of objects, IGLOAD encountered a truncated record.

Unable to do a CONTROL operation on *MSINK*

System error. Contact the Computing Center.

Unable to do a SENSE operation on *MSINK*

System error. Contact the Computing Center.

Unable to get a FDUB on *MSINK* so cannot load DDR

System error. Contact the Computing Center.

Unable to load file "ETC.:IG.FONTS(kkkkkk,llllll)"

A call to IGATTS or IGTXT specified a character set or font name which is legal but for which there is no data.

Unable to load IG-ddrname interface

System error. Contact the Computing Center.

Unexpected end-of-file during read by IGLOAD

In reloading a set of objects, IGLOAD encountered an end-of-file before reaching the end of one of the objects.

Unrecognizable color "cccccc"

The second parameter to the (obsolete) subroutine IGHUE was cccccc instead of one of the legal color names.

Unrecognizable USER command "uuuuuu"

The second parameter to IGUSER was not 'GET ' or 'PUT '. Note that each of these keywords contains four characters, including the blank.

Unrecognized transformation type "tttt"

Either (1) a type parameter to IGTRAN, IGBGNS, or IGPUTO was tttt instead of one of the legal values listed in the IGTRAN description in Appendix B; or (2) a legal type parameter was not followed by the right number of legal modif parameters.

X (Y, Z) value is an integer;
All such integers will automatically be floated

An X, Y, or Z parameter to IGDA, IGDR, IGMA, IGMR, or IGVEC had a high-order byte of X'00' or X'FF', making it look more like an integer than a floating-point number. After this message has been printed once, all such integers are converted to floating-point numbers without further comment.

XL-XR too small for window transformation
YB-YT too small for window transformation

The 'WIND' transformation type requires divisions by xr-xl and yt-yb to determine the X and Y scalings. If one of these divisions overflows, one of the above messages is printed.

LOCATION OF THE CALL THAT PRODUCED THE ERROR

The second line of the error message gives the location of the call that produced the error. The form of this message depends on whether the user program was compiled with *FTN with the ID option ON (the default), and whether the program was being run with the MTS SYMTAB option ON (the default).

ID	SYMTAB	Error Message
ON	ON	AT STATEMENT sssss IN SECTION nnnnnn
OFF	ON	AT LOCATION rrrrrr IN SECTION nnnnnn
--	OFF	AT LOCATION aaaaaa

Here, nnnnnn is the subroutine in the user program from which the offending call was issued; sssss is the FORTRAN internal statement number as printed on the left edge of a compilation listing; rrrrrr is the relative address within nnnnnn of the call; and aaaaaa is the absolute address of the call.

RECOVERY

The $RESTART command may be used to restart the user program, in which case the call that produced the error will be ignored. This, of course, can potentially generate more errors.

APPENDIX E: CHARACTER SETS AND FONTS

The following is a list of the character sets and fonts currently available in IG:

'STANDARD'	Uppercase ASCII
'7ASCII'	Full 7-Bit ASCII
'SANSERIF.1'	
'SANSERIF.2'	
'SANSERIF.CART'	
'ROMAN.2A'	
'ROMAN.2'	
'ROMAN.3'	
'ITALIC.2A'	
'ITALIC.2'	
'ITALIC.3'	
'SCRIPT.1'	
'SCRIPT.2'	
'GREEK.1'	
'GREEK.2A'	
'GREEK.2'	
'GREEK.CART'	
'GREEK'	
'GOTHIC.ENGLISH'	
'GOTHIC.FRAKTUR'	
'GOTHIC.ITALIAN'	
'CYRILLIC.2'	

A default character set or font may be specified for a subpicture by calling IGATTS with one of the above names. A local character set or font may be specified by calling IGTXT with one of the above names. See the descriptions of IGATTS and IGTXT in Appendix B.

On the following pages are translation tables for the various character sets and fonts. Output characters (the top rows in the tables) are always specified by giving their full 7-bit ASCII equivalents (the bottom rows in the tables). For example, when using the 'GREEK.1' character set, a lowercase "alpha" is specified by giving a lowercase "a". Some character sets and fonts do not have equivalents for all of the 7-bit ASCII characters. In such cases, the extra 7-bit ASCII characters are translated into blanks.

Note: In some of the following tables, the spacing between the letters is uneven. This is because variable-width letters have been forced into fixed-width spaces. In actual use, the spacing between the letters will be even.

December 1980

'STANDARD'

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z { | } ~
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

'7ASCII'

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

December 1980

'SANSERIF.1'

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ¬
◙ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! " # $ % ' ( ) * + , − . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ¬
◙ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

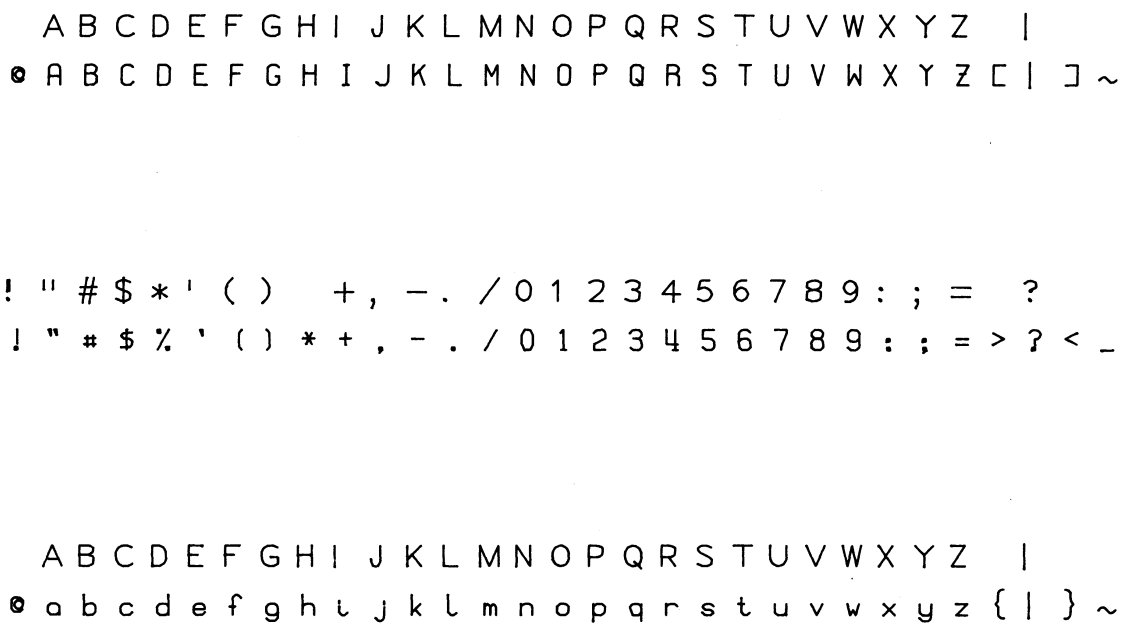
December 1980

'SANSERIF.2'

```
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z      ¬
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ | ] ~

! " $ ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = ? _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

  a b c d e f g h i j k l m n o p q r s t u v w x y z      ¬
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

December 1980

'SANSERIF.CART'



December 1980

'ROMAN.2A'

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|]
◙ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ⌈ | ⌋ ~

! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? <
¡ " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | }
◙ a b c d e f g h ι j k l m n o p q r s t u v w x y z { | } ~

December 1980

'ROMAN.2'

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ¬
◎ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ⊏ | ⊐ ~

! " # $ % ' ( ) * + , − . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ¬
◎ a b c d e f g h ι j k l m n o p q r s t u v w x y z { | } ~

December 1980

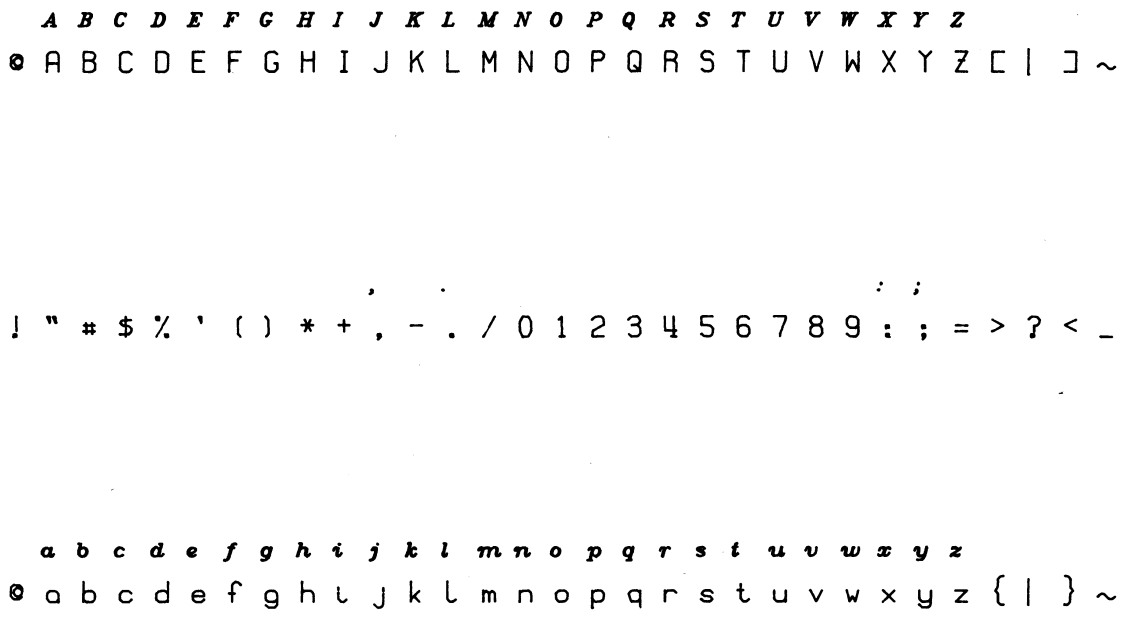
'ROMAN.3'

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! " $ ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = ?
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

a b c d e f g h i j k l m n o p q r s t u v w x y z
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

'ITALIC.2A'


A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
a b c d e f g h i j k l m n o p q r s t u v w x y z
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

December 1980

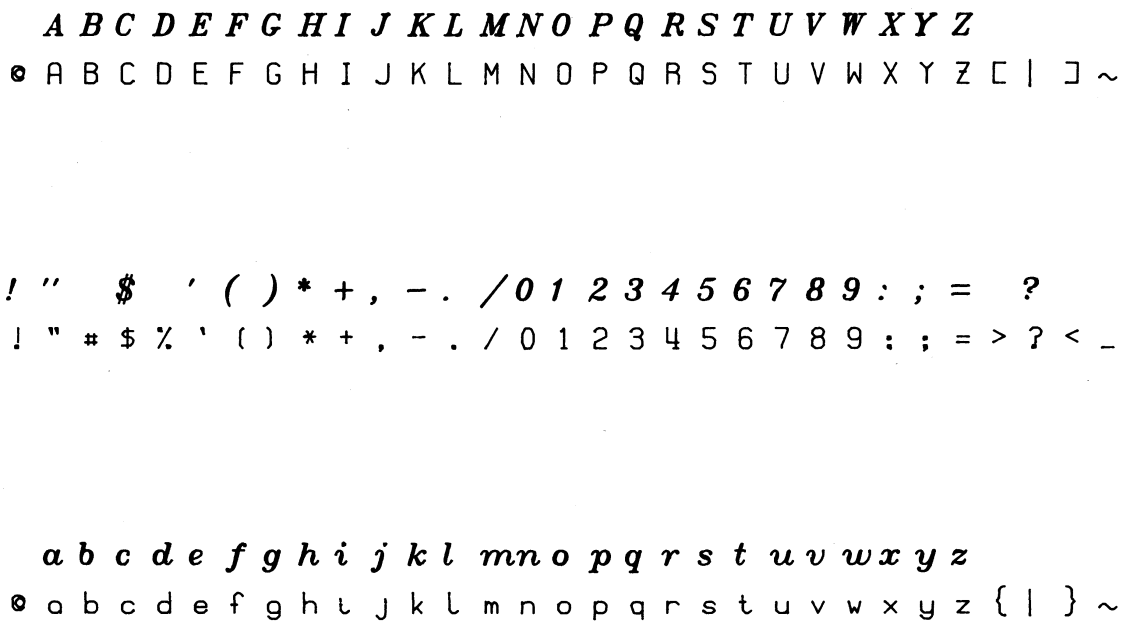
'ITALIC.2'

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ¬
◙ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z ⊏ | ⊐ ~

*! " $ ' ( ) * + , − . 0 1 2 3 4 5 6 7 8 9 : ; = ? _*
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

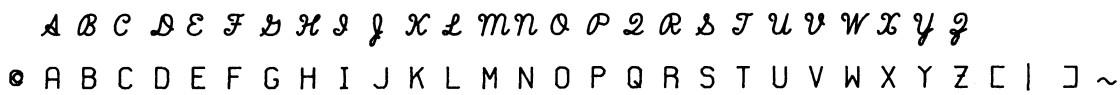
a b c d e f g h i j k l m n o p q r s t u v w x y z ¬
◙ a b c d e f g h ι j k l m n o p q r s t u v w x y z { | } ~

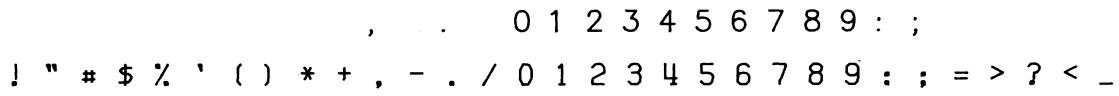
'ITALIC.3'

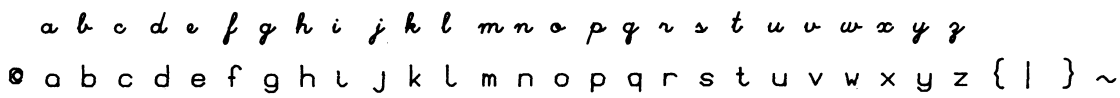

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
! " $ ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = ?
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
a b c d e f g h i j k l m n o p q r s t u v w x y z
a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

December 1980

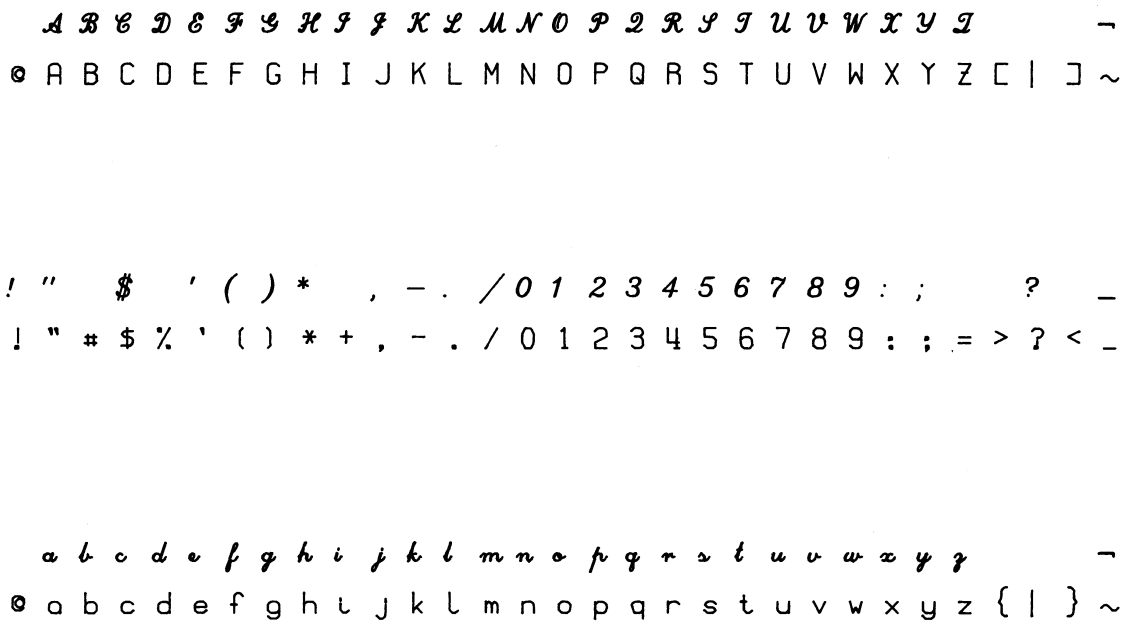
'SCRIPT.1'







'SCRIPT.2'


A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
© A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~
! " $ ' ( ) * , - . / 0 1 2 3 4 5 6 7 8 9 : ; ? _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
a b c d e f g h i j k l m n o p q r s t u v w x y z
© a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

December 1980

'GREEK.1'

@ A B Γ Δ E Z H Θ I K Λ M N Ξ O Π P Σ T Υ Φ X Ψ Ω [|] ¬
◎ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! " # $ % ' ( ) * + , − . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ α β γ δ ε ζ η ϑ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω { | } ¬
◎ a b c d e f g h ι j k l m n o p q r s t u v w x y z { | } ~

December 1980

'GREEK.2A'

```
@ A B Γ Δ E Z H Θ I  K Λ M N Ξ O Π P Σ T Υ Φ X Ψ Ω  [ | ]
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ | ] ~

! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? <
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ α β γ δ ε ζ η ϑ ι  κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω  { | }
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

December 1980

'GREEK.2'

@ Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω [|] ¬
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! " # $ % ' ( ) * + , − . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

@ α β γ δ ε ζ η ϑ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω { | } ¬
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

December 1980

'GREEK.CART'

```
  A B Γ Δ E Z H Θ I   K Λ M N Ξ O Π P Σ T Υ Φ X Ψ Ω    |
◙ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ | ] ~

! " # $ * ' ( )   + , — . / 0 1 2 3 4 5 6 7 8 9 : ; =  ?
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

  A B Γ Δ E Z H Θ I   K Λ M N Ξ O Π P Σ T Υ Φ X Ψ Ω    |
◙ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

December 1980

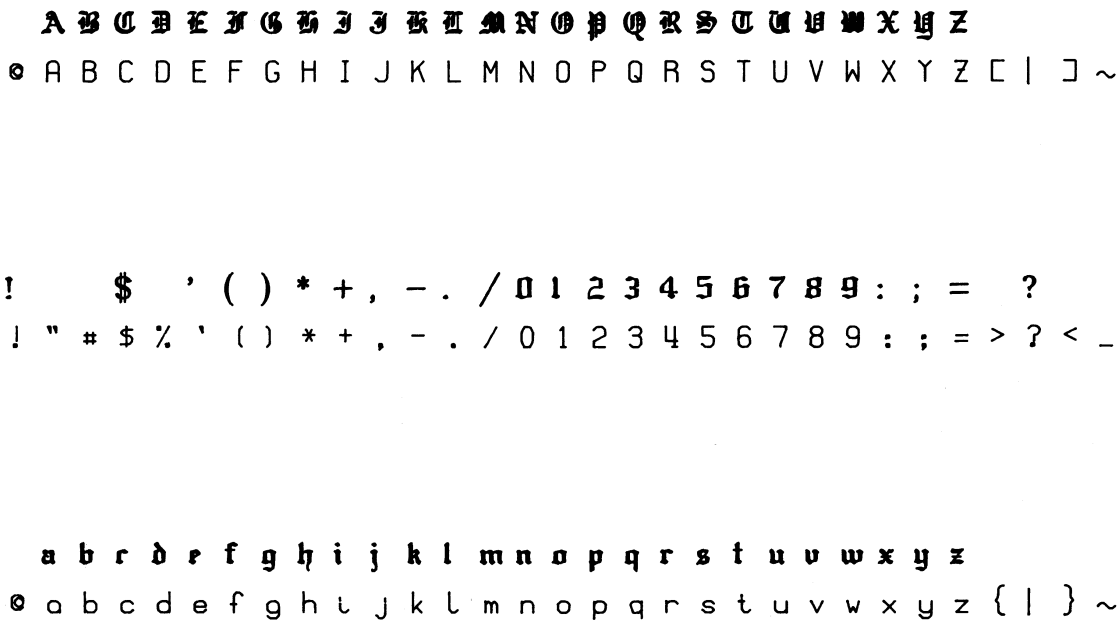
'GREEK'

```
⌀ Α Β Η Δ Ε Φ Γ Χ Ι  Κ Λ Μ Ν Ο Π Θ Ρ Σ Τ Υ  Ω Ξ Ψ Ζ [ | ] ~
⌀ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ | ] ~

! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

⌀ α β η δ ε φ γ χ ι  κ λ μ ν ο π θ ρ σ τ υ  ω ξ ψ ζ { | } ~
⌀ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```

Note: This Greek character set is different from the other Greek character sets, in that it has a different mapping of 7-bit ASCII characters into Greek characters. In most cases, it is preferable to use one of the other Greek character sets.

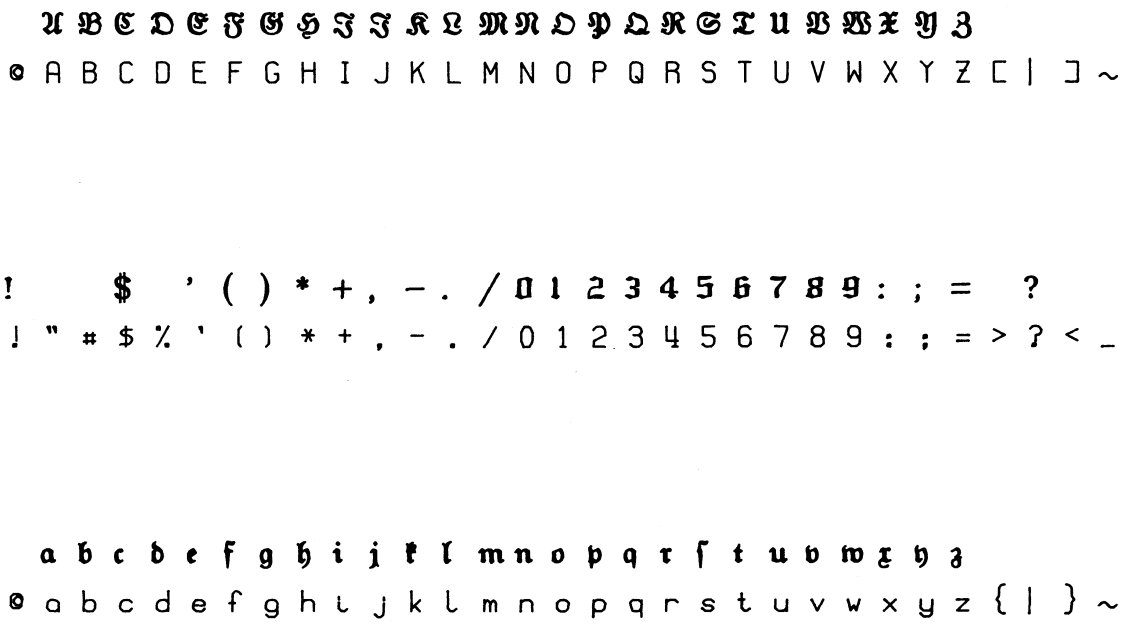
December 1980

'GOTHIC.ENGLISH'



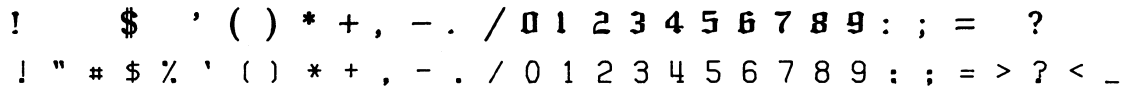
December 1980

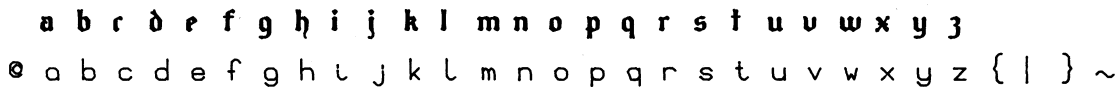
'GOTHIC.FRAKTUR'



December 1980

'GOTHIC.ITALIAN'





December 1980

'CYRILLIC.2'

Б В Г Д Е Ж З И Й М Н О П Р С Т У Ф Ш Щ Ъ Ы Ь Э Ю Я
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [|] ~

! , . 0 1 2 3 4 5 6 7 8 9 : ; ? _
! " # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = > ? < _

б в г д е ж з и й м н о п р с т у ф ш щ ъ ы ь э ю я
@ a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~

December 1980

APPENDIX F: GLOSSARY

active subpicture/object

Initially, the main picture '*MP*'. Otherwise, the subpicture/object that was the parameter of the most recent call to IGBGNS/IGCTNS/IGBGNO.

New lines and text may be added by calling picture-description subroutines. New lower-level subpictures may be added by calling IGBGNS or IGPUTO.

answerback

On a terminal, a small storage unit that contains characters identifying the terminal type.

'*AP*'

An external name that can be used to refer to the active subpicture, provided there is an active subpicture. If an object is active, '*AP*' is undefined.

aspect ratio

A device-dependent parameter giving the ratio of the screen's Y dimension to its X dimension. Equal to the Y coordinate of the top edge of the screen.

attribute

A characteristic associated with a subpicture, controlling the way in which the subpicture is mapped onto the screen for viewing. May be one of the following: transformation, viewport, pen number, default text scale, or default character set or font.

auxiliary device

A device that may be referenced by a keyword previously defined by the user program in a call to IGAUXD. This call also specifies the DDR to be used for graphic I/O operations.

clipping volume

A three-dimensional volume outside of which lines and text are clipped before they are projected. For a perspective projection, this is a truncated pyramid.

contents

The lines, text, and lower-level subpictures belonging to a subpicture or object. The contents of a subpicture are the same as the contents of its object.

current position

A point in the coordinate space of each subpicture/object that represents a "pen" that may be moved around to "draw" the lines and text that make up the contents of the subpicture/object. Each new line or text string originates at the current position. The endpoint of the line or text string then becomes the new current position.

data structure

See "IG data structure".

DDR

See "device-dependent routine".

default character set or font

One of the attributes associated with a subpicture. Normally uppercase ASCII, but may be changed to some other character set or font by calling IGATTS. This character set or font will be used for each text string added by calling IGTXT, IGFMT, or IGSYM, unless some other character set or font has been locally specified by calling IGTXT.

default text scale

One of the attributes associated with a subpicture. Normally the hardware text scale, but may be changed to some other scale (expressed in subpicture coordinates) by calling IGATTS. This scale will be used for each text string added by calling IGTXT or IGFMT, unless some other scale has been locally specified by calling IGTXT.

device-dependent routine (DDR)

An IG routine associated with a specific device type. Resides in the file *IG.devnam, where "devnam" is a the device name. Translates the IG data structure into the hardware commands necessary to generate the screen image or plot. Reads graphic input.

device name

A name used by IG to refer to a specific device type. If "devnam" is a device name, then the associated DDR resides in the file *IG.devnam.

device recognition

A procedure in which IG attempts to determine the device type on which the user is signed on. IG then loads the appropriate DDR.

external name

An optional, user-specified, four-character name for a subpicture or object.

full 7-bit ASCII

One of the character sets supported by IG. Consists of the uppercase ASCII characters plus the lowercase letters.

hardware character set

The keyboard character set of a particular terminal. The character set used for nongraphic I/O operations. Also used for unscaled text generated by IGTXT and for all text generated by IGTXTH.

identity transformation

A transformation that maps each coordinate into a coordinate of the same value. The default transformation for each subpicture.

*IG

The public file containing the subroutines that make up the IG system.

IG data structure

The device-independent data structure that is created by making calls to various IG subroutines. Takes the form of a treelike, directed graph. Consists of the main picture '*MP*' and all subpictures. Each (sub)picture has an associated transformation, viewport, and other attributes.

When IGDRON is called, the data structure is translated into the hardware commands necessary to generate the screen image.

Note that objects created by IGBGNO are not considered to be a part of the data structure until they are attached by calling IGPUTO.

instance

A subpicture, considered in relation to its object. This relationship is expressed by saying that the subpicture is an instance of the object. Several subpictures may be instances of the same object.

internal data structure

See "IG data structure".

internal name

A unique, IG-generated name assigned to each subpicture or object. Takes the form of a large, positive integer.

invisible line

A line having a "move" flag. A line that will not be visible but will instead result in a jump between visible parts of the screen image.

keyword

A character string passed as a parameter to an IG subroutine, to control its execution. For mnemonic purposes, keywords may have more than four characters. In most cases, however, only the first four characters will actually be used.

left-handed coordinate system

A three-dimensional, orthonormal coordinate system whose axis directions can be established by the following rule: Hold the left hand with the thumb at right angles, the first finger extended, and the second finger curled toward the palm. The thumb will correspond to the positive X axis, the first finger to the positive Y axis, and the second finger to the positive Z axis.

The three-dimensional coordinate system for each subpicture is left-handed. The XY plane corresponds to the screen, with the positive X axis pointing right and the positive Y axis pointing up. The positive Z axis points into the screen.

line

One of the primitive elements that make up the contents of a subpicture/object. Represented by the coordinates of its endpoints in the coordinate space of the subpicture/object. May be either visible (having a "draw" flag) or invisible (having a "move" flag).

literal string

In FORTRAN, a constant character string delimited by primes.

locator

A tracking cross, cursor, pair of crosshairs, or some other mechanism that can be moved to any point on the terminal screen, for the purpose of performing graphic coordinate input. The locator coordinates may be read by calling IGXYIN.

main picture

The highest-level picture in the IG data structure. Automatically created, opened, and activated by calling IGINIT. Has the external name '*MP*'.

'*MP*'

The external name for the main picture.

nesting level

The level of a subpicture in the hierarchy of subpictures that make up the IG data structure. See "nesting-level number".

nesting-level number

A number associated with the nesting level of a subpicture. May be established by counting the number of branches that lead back to a subpicture of the main picture. Thus, subpictures of the main picture have nesting level 0, subpictures of these subpictures have nesting level 1, and so on. Successively higher nesting-level numbers represent successively lower nesting levels.

object

A basic data unit. Has an internal name and, optionally, an external name. Contains lines, text, and lower-level subpictures. May be attached to a subpicture, in which case the contents of the subpicture are the same as the contents of the object.

open subpicture/object

Any subpicture/object for which IGBGNS/IGCTNS/IGBGNO has been called but IGENDS/IGENDO has not yet been called. See "stack of open subpictures/objects".

orthographic projection

A projection of a three-dimensional figure onto a two-dimensional plane, such that the projection lines are all parallel and perpendicular to the plane.

In the process of mapping the IG data structure into screen coordinates, an orthographic projection is applied to each three-dimensional subpicture. This is done by merely ignoring the Z coordinates. Thus, the subpicture is mapped into the XY plane.

This orthographic projection may be replaced by a perspective projection by calling IGTRAN.

PDS

See "Plot Description System".

PDS file

A file containing one or more plot descriptions, each of which consists of instructions that can be used to drive the CalComp plotter.

pen number

One of the attributes associated with a subpicture. This number is a code that represents various line qualities that can be produced by the output device (using various "pens"). For example, a given pen number might represent a specific hue or intensity level. The actual interpretation of a pen number will depend on the type of output device.

perspective projection

A projection of a three-dimensional figure onto a two-dimensional plane, such that the projection lines converge on a single viewpoint (representing the eye of the observer).

In IG, a perspective projection may be specified for a subpicture by calling IGTRAN. The viewpoint will be on the -Z axis. The subpicture will be projected onto the XY plane. Any lines extending in front of the XY plane (into the half-space Z<0) will be clipped, and any lines extending beyond the maximum visible Z coordinate (beyond the plane Z=F) will be clipped.

pick locator

A light pen, cursor, pair of crosshairs, or some other mechanism that can be moved to any point on the terminal screen, for the purpose of picking a subpicture by pointing to it on the screen. Depending on the type of terminal, the pick locator may or may not be the same mechanism as the locator that is used for graphic coordinate input. See "locator".

picture

The main picture '*MP*' or any subpicture. See "subpicture".

picture-description subroutine

An IG subroutine that is used to add primitive elements (lines or text strings) to the contents of the active subpicture/object.

plot description

A collection of instructions that can be used to drive the CalComp plotter.

Plot Description System (PDS)

A subroutine package for generating plot descriptions, which may in turn be used to drive the CalComp plotter. PDS resides in the public file *PLOTSYS.

*PLOTSYS

The public file containing the Plot Description System (PDS).

PLTBGN

The PDS subroutine that is called to begin a plot description. This subroutine need not be called explicitly by the user program. The first call to any other PDS subroutine generates an implicit call to PLTBGN.

IG has its own version of this subroutine, thus allowing any program that calls PDS subroutines to interface with IG.

PLTEND

The PDS subroutine that is called to end a plot description.

IG has its own version of this subroutine, thus allowing any program that calls PDS subroutines to interface with IG.

queue of plot requests

An MTS queue that contains requests for the generation of CalComp plots from specific PDS files.

Requests may be entered or deleted by means of the public file program *CCQUEUE.

raster-refresh-type display

A television display that is continually refreshed from an associated memory unit.

rotation

A type of transformation in which a subpicture is rotated about its X, Y, or Z axis.

scaling

A type of transformation in which a subpicture is magnified or reduced through multiplication by a scalar.

screen coordinates

A two-dimensional, orthonormal coordinate system corresponding to the terminal screen. The origin (0,0) corresponds to the center of the screen, the point (+1,0) corresponds to the extreme right edge of the screen, and the point (-1,0) corresponds to the extreme left edge of the screen. The Y coordinates (vertical axis) have the same units as the X coordinates (horizontal axis).

When IGDRON is called, the IG data structure is mapped into screen coordinates.

screen image

The image on the terminal screen resulting from a call to IGDRON.

selective-erase capability

A characteristic of some terminals that allows them to selectively erase parts of the screen without having to regenerate the entire image. This can speed up small modifications to the screen image.

stack of open subpictures/objects

A push-down stack containing the names of all open but currently inactive subpictures/objects.

When IGBGNS/IGCTNS/IGBGNO is called for a subpicture/object, the currently active subpicture/object is deactivated and pushed onto the stack. Then, the new subpicture/object is activated.

When IGENDS/IGENDO is called for the currently active subpicture/object, it is deactivated and closed. Then, the previously active subpicture/object is popped from the stack and reactivated.

storage-tube-type terminal

A terminal in which an electron beam is used as a "pen" to "write" lines and text on the screen. The resulting image is stored (as opposed to being continually refreshed).

subpicture

A basic data unit, corresponding to a branch in the IG data structure. Has an internal name and, optionally, an external name. Has an associated set of attributes (transformation, viewport, and others) that determine the way in which it is to be mapped onto the screen for viewing. Has an object that may contain lines, text, and/or lower-level subpictures.

Synonymous with "picture".

subpicture coordinates

A three-dimensional, left-handed, orthonormal coordinate system associated with each subpicture. The lines and text that make up the contents of the subpicture are defined within this coordinate space.

The transformation associated with the subpicture is used to map its contents into the coordinate space of the next-higher-level picture.

text string

One of the primitive elements that make up the contents of a subpicture/object.

transformation

One of the attributes associated with a subpicture. A linear transformation that is used to map the contents of the subpicture into the coordinate space of the next-higher-level picture.

Represented internally by a 4x4 matrix.

Initially, the identity transformation. May be respecified by calling IGTRAN.

translation

A type of transformation in which a subpicture is shifted in a given direction through the addition of a vector.

uppercase ASCII

The default character set used by IG. The term "ASCII" refers only to the selection of characters included. (Internally, all characters are represented in EBCDIC codes.)

user word

A single word (four bytes), associated with a subpicture, that is set aside for use by the user. Typically used as a pointer to a block of auxiliary information associated with the subpicture.

The contents of the user word may be stored or fetched by calling IGUSER.

vector-refresh-type display

A display that is continually refreshed by a hardware vector generator, which is in turn driven by a display list stored in an associated memory.

viewing distance

In perspective projection, the distance from the viewpoint to the screen.

viewport

One of the attributes of a subpicture. A rectangular subregion of the screen into which the subpicture will be mapped. A "substitute screen".

Initially, the square -1,+1,-1,+1. May be respecified by calling IGVWPT.

visible line

A line having a "draw" flag. A line that will be visible in the screen image (provided it falls within the screen boundaries).

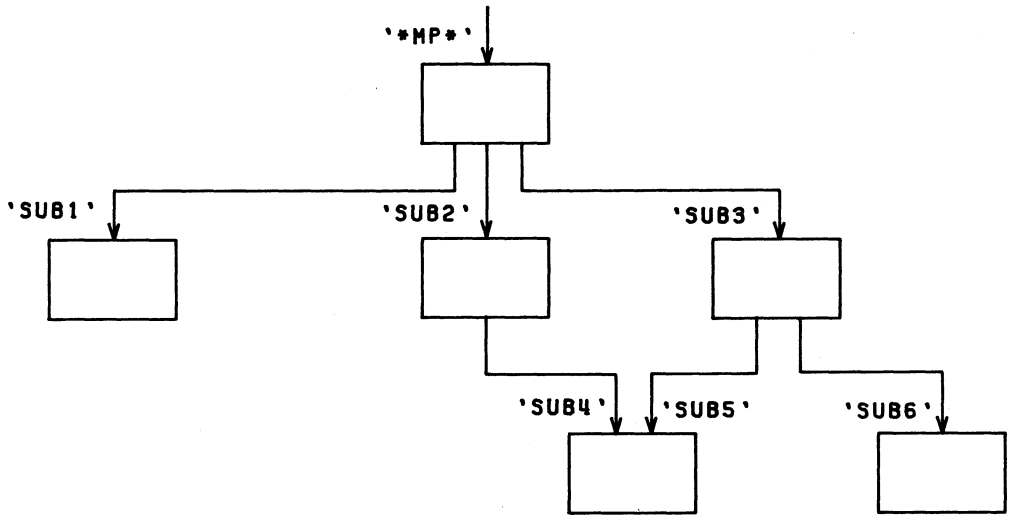
window

A rectangular subregion of the XY plane, in the coordinate space of a given subpicture, that is to be mapped into the viewport associated with the subpicture.

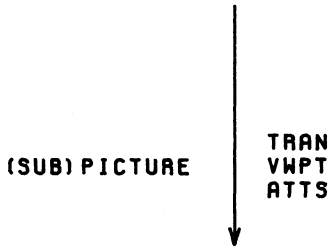
May be specified by calling IGTRAN.

APPENDIX G: IG DATA STRUCTURE

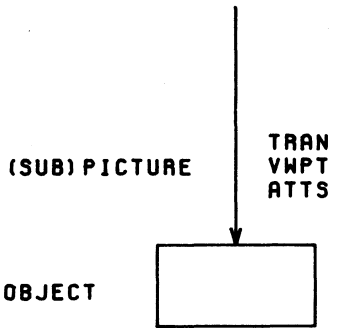
The IG data structure is a treelike, directed graph:



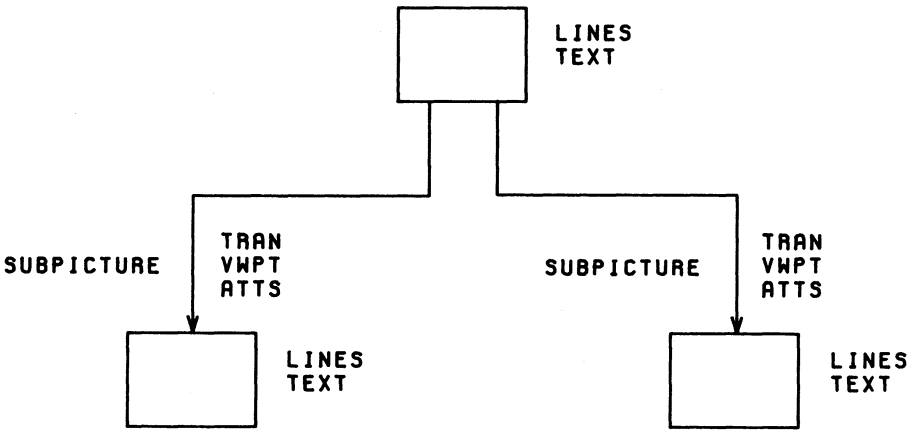
The trunk is the main picture, '*MP*', which is created by calling IGINIT. Each of the other branches is a subpicture, which is created by calling either IGBGNS or IGPUTO. A subpicture may or may not have a user-specified, four-character, external name. In any case, a subpicture will have an IG-generated internal name. Each subpicture has an associated transformation, viewport, and other attributes:



Each subpicture points to a unique object:

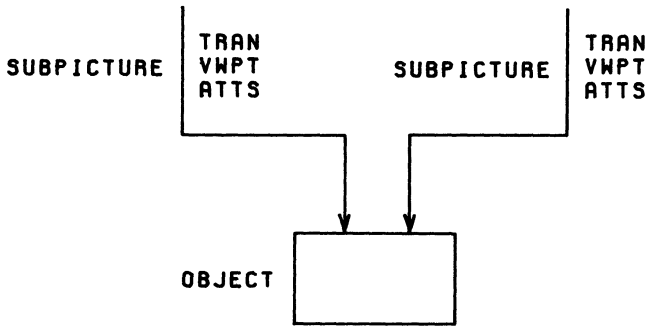


The contents of the subpicture are the same as the contents of the object. These contents may include lines, text and/or lower-level subpictures. Thus, an object will have the following kind of structure:



When a subpicture is created by calling IGBGNS, an object is created implicitly and attached to the subpicture. However, an object may be created explicitly by calling IGBGNO, in which case the object may be referenced by the user. Such an object may or may not have a user-specified, four-character, external name. In either case, it will have an IG-generated internal name.

A user-created object may be attached to the data structure by calling IGPUTO. This creates a subpicture that is an instance of (i.e., points to) the object. Several subpictures may be instances of the same object:



A user-created object is not considered part of the data structure until it is attached by calling IGPUTO. Thus, for example, an unattached object will not be saved when IGDRON('SAVE') is called.

CREATING THE DATA STRUCTURE

Following a call to IGBGNS or IGCTNS, the named subpicture becomes open and remains so until a matching call to IGENDS. Similarly, following a call to IGBGNO, the named object becomes open and remains so until a matching call to IGENDO.

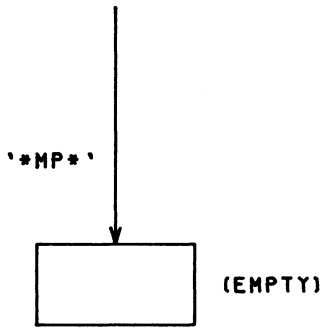
At any given time, exactly one subpicture/object is active. When a call is made to IGBGNS, IGCTNS, or IGBGNO, the currently active subpicture/object is deactivated and pushed onto the stack of open subpictures/objects (or simply, the stack). Then, the subpicture/object named in the subroutine call is activated.

New lines and text may be added to the active subpicture/object by calling the picture-description subroutines. New lower-level subpictures may be created as subpictures of the active subpicture/object by calling IGBGNS or IGPUTO.

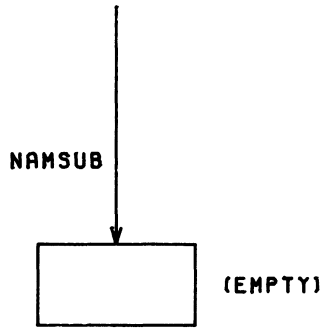
When a call is made to IGENDS or IGENDO, a check is made to ensure that the named subpicture/object is active. Then, this subpicture/object is deactivated and closed. The previously active subpicture/object is popped from the stack and reactivated.

The following paragraphs describe the ways in which the various IG subroutines modify the data structure:

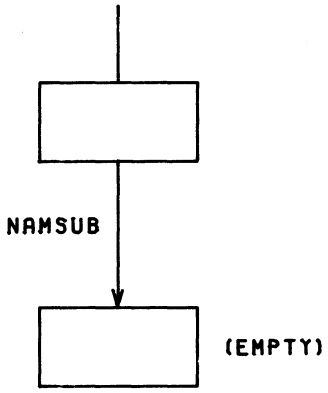
CALL IGINIT creates, opens, and activates the main picture, '*MP*':



CALL IGBGNS(NAMSUB) deactivates the currently active subpicture/object and pushes it onto the stack. The next step depends on whether NAMSUB already exists. If it already exists, all of its lines, text, and subpictures are deleted. It is then opened and reactivated so that its contents may be respecified:



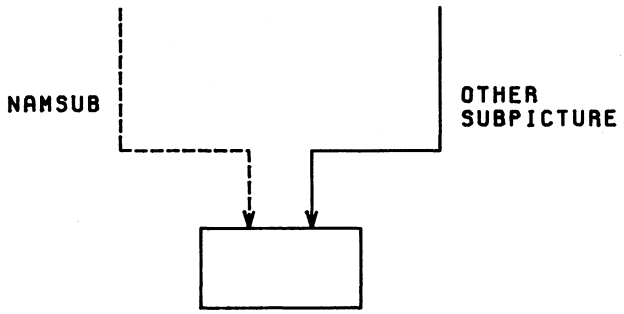
If NAMSUB does not already exist, it is created as a subpicture of the previously active subpicture/object, opened, and activated:



CALL IGCTNS(NAMSUB) deactivates the currently active subpicture/object and pushes it onto the stack. It then opens and reactivates NAMSUB, which is assumed to already exist.

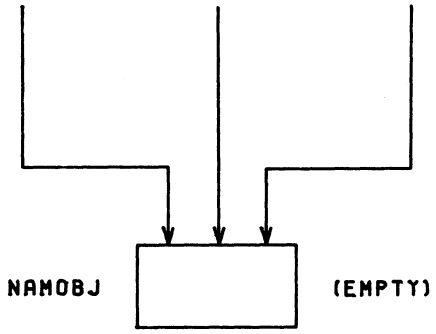
CALL IGENDS(NAMSUB) makes sure that NAMSUB is active, then deactivates and closes it. IGENDS then pops the previously active subpicture/object from the stack and reactivates it.

CALL IGDELS(NAMSUB) deletes NAMSUB from the data structure. Note, however, that if NAMSUB is one of several instances of its object, then its object is not deleted. Only the branch corresponding to NAMSUB is deleted:

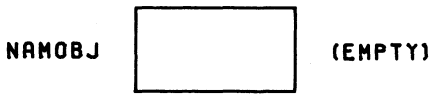


CALL IGBGNO(NAMOBJ) deactivates the currently active subpicture/object and pushes it onto the stack. The next step depends on whether

NAMOBJ already exists. If it does already exist, all of its lines, text, and subpictures are deleted. It is then opened and reactivated so that its contents may be respecified. All subpictures that are instances of NAMOBJ are correspondingly respecified:



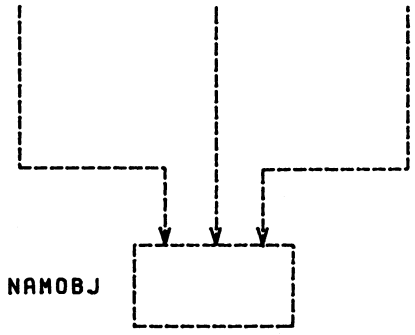
If NAMOBJ does not already exist, it is created, opened, and activated:



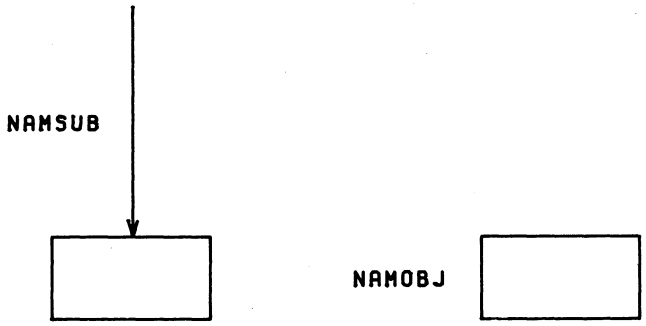
In the latter case, NAMOBJ is a "free-standing" object which is not yet a part of the data structure. It may be attached to the data structure by calling IGPUTO.

CALL IGENDO(NAMOBJ) makes sure that NAMOBJ is active, then deactivates and closes it. IGENDO then pops the previously active subpicture/object from the stack and reactivates it.

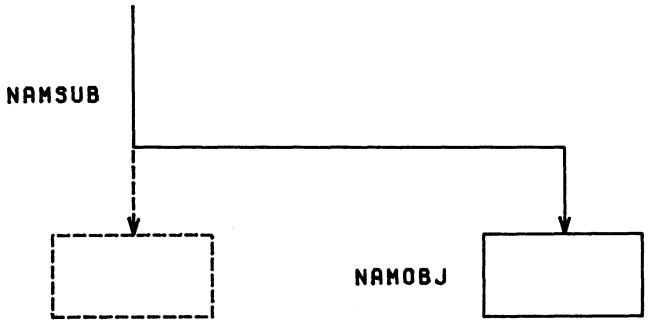
CALL IGDELO(NAMOBJ) deletes NAMOBJ from the data structure, together with all of its instances:



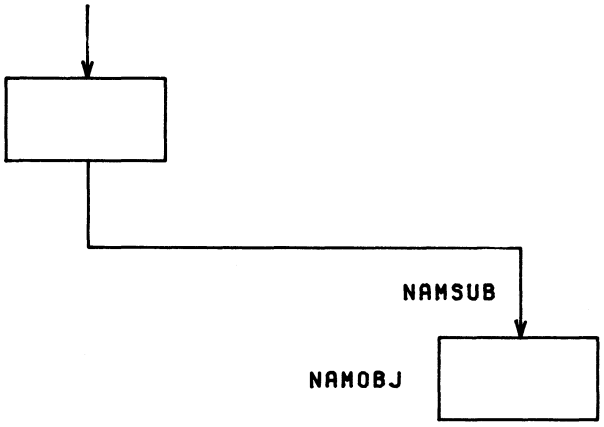
CALL IGPUTO(NAMOBJ,NAMSUB) attaches the (existing) object NAMOBJ to the subpicture NAMSUB. If NAMSUB already exists,



all of its lines, text, and subpictures are deleted and it is respecified as an instance of NAMOBJ:



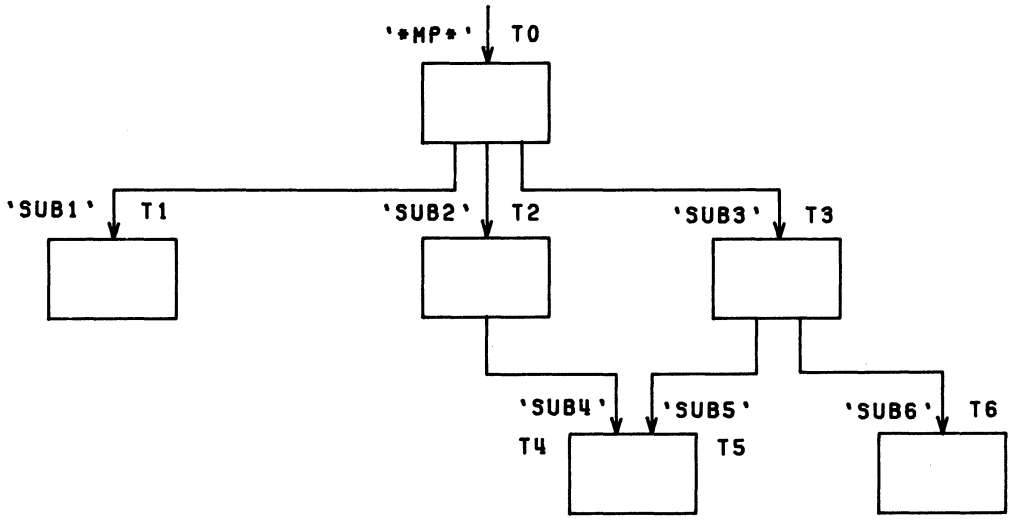
If NAMSUB does not already exist, it is created as a subpicture of the active picture/object, and specified as an instance of NAMOBJ:



TRANSFORMATIONS, VIEWPORTS, AND OTHER ATTRIBUTES

Each subpicture in the data structure has several associated attributes, including a transformation, viewport, pen number, default text scale, and default character set or font. These are independent of the contents of the subpicture and may be specified at any time after the subpicture has been created.

The transformation associated with a subpicture is used to map the contents of the subpicture into the coordinate space of the next-higher-level picture. This transformation is represented internally by a 4x4 matrix. By default, this transformation is the identity transformation, but may be respecified by calling IGTRAN. All transformations are applied at the time IGDRON is called. In the following diagram,



the effective transformation that is used to map the contents of 'SUB4' into screen coordinates is

T4•T2•T0

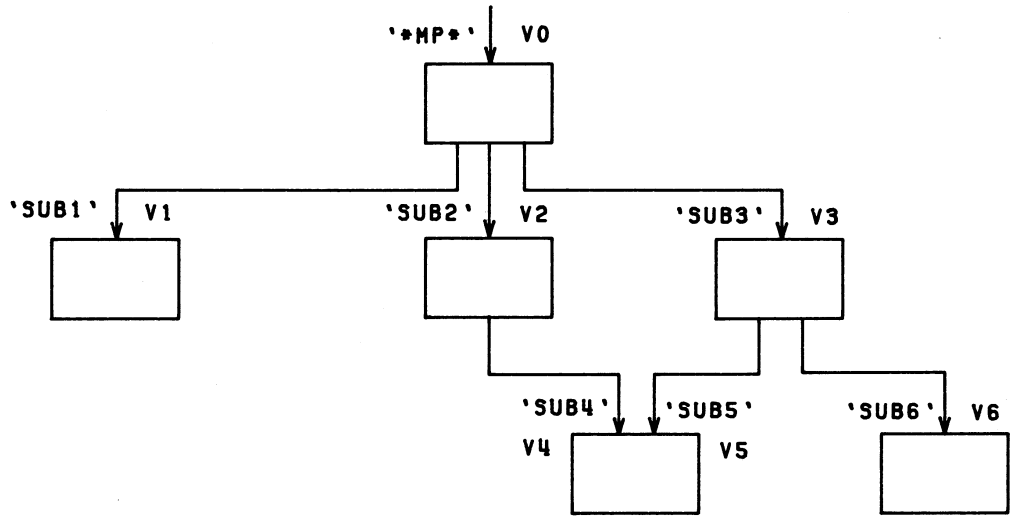
i.e., T4 followed by T2 followed by T0. (Here, the concatenation reads from left to right, in accordance with the notation used by IGTRAN.) Similarly, the effective transformation that is used to map the contents of 'SUB5' into screen coordinates is:

T5•T3•T0

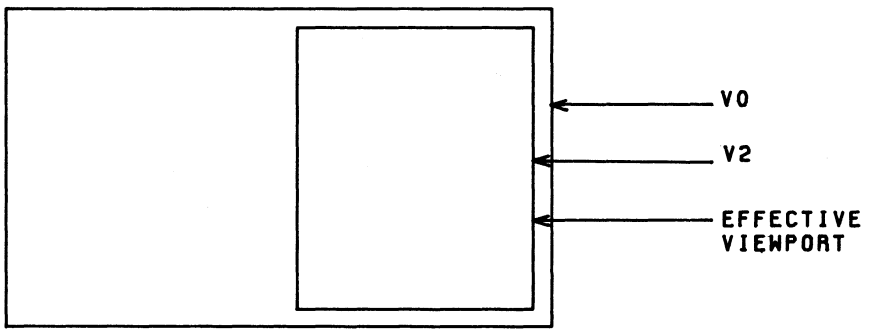
Note that subpictures 'SUB4' and 'SUB5' are instances of the same object. This object is mapped into screen coordinates in two different ways.

The viewport associated with a subpicture is the rectangular sub-region of the screen into which the subpicture is to be mapped. This viewport is always specified in screen coordinates. By default, the viewport is the whole screen, -1,+1,-1,+1, but may be respecified by calling IGVWPT.

The effective viewport that is actually used to display the subpicture consists of the viewport specified by IGVWPT intersected with the effective viewport of the next-higher-level picture. In the following diagram,



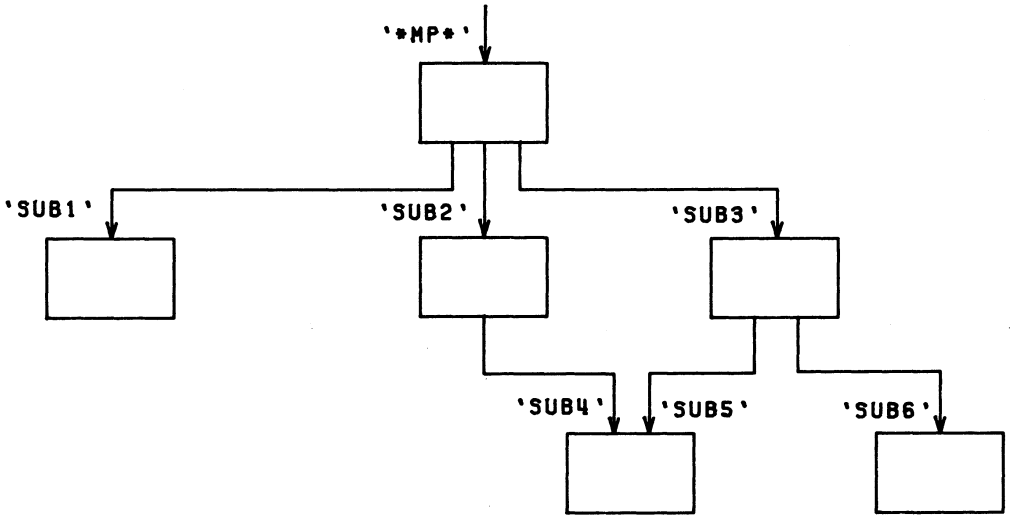
the effective viewport for 'SUB2' is the intersection of V2 and V0 (which in this case is the same as V2):



When IGDRON is called, the transformation T2•T0 is applied to the contents of 'SUB2'. Then, the viewport mapping is applied. In general, the transformation is applied before the viewport mapping.

In most applications, only subpictures of '*MP*' will have viewports specified by IGVWPT. Thus, in the above diagram, V1, V2, and V3 might be specified by IGVWPT, while V0, V4, V5, and V6 would be allowed to default to -1,+1,-1,+1. By the intersection rule, the effective viewport for 'SUB4'and 'SUB5' would be V2 and the effective viewport for 'SUB6' would be V3.

All of the other attributes may be specified by calling IGATTS. The attributes of a given subpicture are inherited by all lower-level subpictures unless explicitly specified otherwise. For example, in the diagram,



the attributes of 'SUB3' will be inherited by 'SUB5' and 'SUB6'. However, different attributes could be specified for 'SUB5' and 'SUB6' by calling IGATTS.

APPENDIX H: GENERATING THE SCREEN IMAGE

Most of the subroutines in the IG system operate on the device-independent IG data structure. The IGDRON subroutine scans this data structure and translates it into the hardware commands necessary to generate the screen image or plot. The IGDRON subroutine accepts the following keyword parameters:

'TERMINAL' The data structure is translated into hardware commands appropriate to the currently active terminal.

'CALCOMP' The data structure is translated into a plot description and written on logical I/O unit 9 or a file or device specified by the user. This plot description may later be used to drive the CalComp plotter (see Appendix K).

'SAVE' The data structure is dumped on logical I/O unit SPUNCH or a file or device specified by the user (see the section "Saving the Current Data Structure as an Object").

The IGDRON subroutine also accepts user-defined keywords for auxiliary devices (see the description of IGAUXD in Appendix B). In the process of translating the data structure into hardware commands, IGDRON performs four kinds of operations:

(1) Incremental updating of the screen image (for the 'TERMINAL' device only). IGDRON checks what changes have been made to the data structure and makes the corresponding changes to the screen image. In this way, it does not always have to regenerate the entire image.

(2) Transformations and clipping.

(3) Simulation of facilities not supported by the hardware.

(4) Data filtering.

The following sections describe these operations in more detail, so the user can tell what to expect as the result of a given call to IGDRON.

INCREMENTAL UPDATING

Some terminals have a selective-erase capability, making it possible to delete part of the screen image without having to regenerate the whole thing. When IGDRON is called, it checks what operations have been performed on the IG data structure since the last call to IGDRON. These fall into three categories:

(1) Operations which result in additions to the screen image. For example, IGDA and IGVEC may add lines, or IGPUTO may add a new instance of an object.

(2) Operations which result in both deletions from and additions to the screen image. As an example, if IGTRAN causes the movement of a subpicture, then the original screen image of the subpicture must be deleted and the new screen image of the subpicture must be added. As another example, if IGATTS is used to change the pen number (hue or intensity) of a subpicture, then a similar deletion and addition are necessary.

(3) Operations which result in deletions from the screen image. Some examples are IGDELS, IGBGNS (if the subpicture already exists), IGDELO, and so on.

For terminals with a selective-erase capability, IGDRON has the choice of either erasing the entire screen and regenerating the entire image or selectively erasing parts of the screen. Whichever action requires the least work is the one that is performed.

For terminals without a selective-erase capability, IGDRON normally erases the entire screen and regenerates the entire image. This default mode of operation is known as the automatic-erase mode. However, a keep mode is also available, in which IGDRON does not erase the screen. This mode of operation may be entered by calling IGCTRL as follows:

```
CALL IGCTRL('TERMINAL','KEEP',1)
```

In keep mode, IGDRON makes only the necessary additions to the image. The parts that would normally have been deleted are allowed instead to remain on the screen. Thus, the screen image no longer reflects the current data structure. To bring the image up to date, the user may make the following calls:

```
CALL IGCTRL('TERMINAL','ERASE')
CALL IGDRON('TERMINAL')
```

The first call erases the screen, and the second regenerates the entire image. The automatic-erase mode of operation may be restored by making the following call:

```
CALL IGCTRL('TERMINAL','KEEP',0)
```

TRANSFORMATIONS AND CLIPPING

All subpictures in the IG data structure have associated transformations (specified by IGTRAN) and viewports (specified by IGVWPT). When IGDRON is called, it uses these to map the subpictures into -1 to +1 screen coordinates. At this time, it detects some special-case transformations and, whenever possible, shortens the mapping process. In addition, it applies a device-dependent transformation (which the user never sees directly) to map the -1 to +1 screen coordinates into the integer raster-point coordinates of the output device.

SIMULATION OF FACILITIES NOT SUPPORTED BY HARDWARE

Different output devices have different capabilities for generating hardware text. Some (such as the CalComp plotter) have no hardware text capabilities at all. Before displaying a text string on the screen, IGDRON checks the scale and rotation angle of the text string against the hardware text capabilities of the output device. Whenever possible, IGDRON uses a hardware text generator to display the text string. Otherwise, IGDRON calls a software character generator to convert the text string into a series of line segments, which are then transformed appropriately and displayed on the screen.

Different output devices also have different chromatic capabilities (see Appendix I). A given device might support several different hues or several different line intensities (weights).

Each subpicture in the IG data structure has a pen number (specified by a call to IGATTS). When IGDRON is called, the pen numbers are mapped into the hues/intensities available on the device. For example, the pen numbers 1, 2, and 3 might be mapped into the hues black, red, and blue. Pen numbers outside of this range will also be mapped into the range of available hues. In this way, the same data structure might be displayed in different hues/intensities on different output devices.

DATA FILTERING

After the lines of the data structure have been transformed and mapped into the integer raster-point coordinate system of the output device, various "null graphical operations" are eliminated. First, all consecutive invisible lines (moves) are merged into one invisible line. Then all consecutive visible lines which are colinear are merged into one visible line. In particular, any zero-length visible lines are deleted. The number of lines actually displayed may therefore be less than the number of lines in the data structure, especially if the data structure contains very short or scaled-down lines. This filtering process minimizes the number of hardware commands necessary to generate the screen image, thus reducing transmission time and reducing the chance of burning the phosphor on the screen.

APPENDIX I: DEVICES SUPPORTED BY IG

DEVICE-DEPENDENT ROUTINES (DDRS)

User programs may perform graphic output by calling the IGDRON subroutine:

```
CALL IGDRON(device)
```

Here, "device" may be one of the keywords 'TERMINAL', 'CALCOMP', or 'SAVE', or a user-defined keyword for an auxiliary device. Depending on the "device" keyword, IGDRON will call the appropriate device-dependent routine (DDR) to generate the output:

user-defined	The DDR specified by the user program in the call to IGAUXD.
'SAVE'	The object-saving DDR located in the file *IG.SAVE.
'CALCOMP'	The CalComp DDR located in the file *IG.CCMP.
'TERMINAL'	The DDR loaded by the user at $RUN time or by IG following its device-recognition procedure.

Note that 'SAVE' and 'CALCOMP' are always associated with the same DDRs. On the other hand, 'TERMINAL' may be associated with a variety of DDRs. In most cases, the user will allow the 'TERMINAL' DDR to be automatically loaded by IG. However, the user may explicitly load the 'TERMINAL' DDR at $RUN time, as follows:

```
$RUN objectfile+*IG+*IG.devnam
```

The file *IG.devnam contains a DDR, normally the DDR for the terminal on which the user is signed on. For example,

```
$RUN OBJ+*IG+*IG.MX12000
```

loads the DDR for the Magnavox 12000 terminal. However, when running from a nongraphics terminal or from batch, the user may wish to direct the 'TERMINAL' output either to the CalComp plotter,

```
$RUN OBJ+*IG+*IG.CCMP
```

or to *MSINK* in the form of printer-plot output:

```
$RUN OBJ+*IG+*IG.PRNT
```

In the latter case, each call to IGDRON('TERMINAL') writes a square plot in the form of one page of printer output. Various printer characters are used to simulate the shapes of the lines. The resolution of this plot is rather low: 100 characters by 60 lines.

DEVICE RECOGNITION

If the 'TERMINAL' DDR is not explicitly loaded by the user at $RUN time, IG performs the following device-recognition procedure:

(1) IG makes an attempt to obtain the device type from the network. This will succeed if the network knows the type of terminal either through a default setting or through having received a TERMINAL device command. If this fails,

(2) IG makes an attempt to obtain the device type by polling the terminal. This will succeed if the terminal has an appropriate answerback or other distinguishing characteristics. If this fails,

(3) IG asks the user for the device name. (The device name is "devnam" if and only if the associated DDR resides in the file *IG.devnam.) If the user gives an illegal device name, IG will list the legal device names and ask again.

Note that in most cases, the device-recognition procedure will be automatic, i.e., step (3) will not be reached. For more details, see the individual device descriptions below.

In batch mode, the above procedure does not apply. If the 'TERMINAL' DDR is not explicitly loaded by the user at $RUN time, IG loads the nongraphics terminals DDR located in the file *IG.TTY. In batch mode, graphic input operations are illegal.

INDIVIDUAL DEVICE DESCRIPTIONS

The remainder of this appendix consists of individual device descriptions, each of which gives the device-recognition procedure and the interaction of the device with the subroutines IGDRON, IGXYIN, IGPIKS, and IGCTRL.

December 1980

Anderson-Jacobson 830 Terminal with Plot Option

General Features:

The Anderson-Jacobson 830 is a hard-copy terminal with a Diablo-type print mechanism. It operates in either normal mode, in which it prints character output, or graphics mode, in which it prints plotter output in the form of small dots. In graphics mode, it generates 10-inch square plots having 600 x 480 visible points. It offers a choice of two colors, red and black, for graphic output. The AJ830 has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the AJ830 resides in the file *IG.AJ830.

Device Recognition and Setup:

An AJ830 is automatically recognized by IG if it has an answerback of "AJ830" or the terminal type has been set to "AJ830" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). The parity switch should be set to NONE or plotting will not work properly.

Graphic Output:

Each call to IGDRON regenerates the entire plot.

Graphic Output Attributes:

The following pen numbers are available and may be selected for a given subpicture by calling IGATTS:

1 black
2 red

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any

invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','RESET')

Resets the terminal from graphics mode to normal mode (e.g., following an attention interrupt).

CalComp Plotter

General Features:

The CalComp plotter is the general system plotter. The maximum size of the plotter paper is 360 inches in the X dimension and 33 inches in the Y dimension. The plotter has four different pens with four different colors of ink: black, red, blue, and green.

Device-Dependent Routine:

The DDR for the CalComp plotter resides in the file *IG.CCMP.

Device Recognition and Setup:

The CalComp plotter is always accessible as the 'CALCOMP' device. However, the user may also wish to use the CalComp plotter as the 'TERMINAL' device (e.g., when running from a nongraphics terminal or from batch). To do this, the user must explicitly load the CalComp DDR at $RUN time by concatenating *IG+*IG.CCMP to the object file.

Graphic Output:

Output may be generated for the CalComp plotter by making a call to IGDRON('CALCOMP').

Each call to IGDRON('CALCOMP') writes a complete plot description onto logical I/O unit 9 or onto a file specified by a call to IGCTRL('CALCOMP','PFILE','pdsfile '). This plot description may later be used to generate an actual CalComp plot. To do this, the user must $RUN *CCQUEUE to enter a plot request into a special system queue (see Appendix K).

The plot will be similar to the screen image that would have been generated by a call to IGDRON('TERMINAL'). Note, however, that if the terminal has a rectangular screen, then the top and bottom of the screen image will be clipped, while the corresponding parts of the plot will not.

By default, the plot is scaled to fit a 7.5x7.5-inch square centered in an 8.5x11-inch sheet of plotter paper. This scaling and placement can be changed by a call to IGCTRL.

When the CalComp plotter is also in use as the 'TERMINAL' device, output may be generated for it by making a call to

IGDRON('TERMINAL'). Each call writes a complete plot description onto logical I/O unit 9 or onto a file specified by a call to IGCTRL('TERMINAL','PFILE','pdsfile ').

Graphic Output Attributes:

The following pen numbers are available and may be selected for a given subpicture by calling IGATTS:

1	black
2	red
3	blue
4	green

Graphic Input:

Graphic input operations are not performed with the 'CALCOMP' device. The following paragraphs apply only when the CalComp plotter is in use as the 'TERMINAL' device.

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('CALCOMP','SIZE',S)

Sets the plot size to S inches. The default value is 7.5 inches.

IGCTRL('CALCOMP','XMAR',XMARGN)

Sets the X margin of the plot to XMARGN. The default value is 0.5 inches.

IGCTRL('CALCOMP','YMAR',YMARGN)

Sets the Y margin of the plot to YMARGN. The default value is 1.75 inches.

IGCTRL('CALCOMP','PFILE','pdsfile ')

Assigns "pdsfile" as the file onto which subsequent plot descriptions will be written. Note that the file name must be followed by a trailing blank.

IGCTRL('CALCOMP','PHDR',0)

Turns off the message PDS: PLOT DESCRIPTION GENERATION BEGINS. Once this message has been turned off, it may not be turned back on.

When the CalComp plotter is also being used as the 'TERMINAL' device, the above control operations may also be applied to 'TERMINAL'.

Computek 400 Terminal

General Features:

The Computek 400 is a storage-tube-type terminal. It has an 8.25x6.4-inch rectangular screen with 1024 x 801 visible points. It operates in either normal mode, in which it reads keyboard input and writes character output, or graphics mode, in which it does graphic I/O. The CK400 has a cursor that can be moved about the screen by pressing the "arrow" buttons that surround the HOME button. This cursor is used for both graphic coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the CK400 resides in the file *IG.CK400.

Device Recognition and Setup:

A CK400 is automatically recognized if it is connected through the UMnet Computer Network at 300 baud or through an RFL or Vadic dataset. A CK400 is automatically recognized through any terminal control unit if the the terminal type has been set to "C400" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). When using this type of terminal with the UMnet Computer Network, it is not possible or advisable to queue input text ahead of the program during device recognition or before any IGDRON call that does not erase the screen. This input queueing interferes with the implicit cursor-read operation performed at these times. Also note that input text (other than CTRL-E) is ignored while the screen image is being generated.

Graphic Output:

The IGDRON subroutine bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the screen image. If lines have been moved or deleted, the screen will be erased and the entire image regenerated.

Graphic Input:

The IGXYIN subroutine first positions the cursor at the coordinates of the last graphic input operation (or at the center of the screen, if this is the first operation). The user may then move the cursor by pressing the "arrow" buttons that surround the HOME button. The cursor position may be returned by hitting any keyboard key. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine first positions the cursor at the coordinates of the last graphic input operation. The user may then move the cursor to pick a subpicture. The cursor position may be returned by hitting any keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','RESET')

Resets the terminal from graphics mode to normal mode (e.g., following an attention interrupt).

December 1980

Data Terminals Corporation 300 or 302 Terminals

DTC300
GSI300
DTC302

General Features:

The DTC300, GSI300, and DTC302 are hard-copy terminals with Diablo-type print mechanisms. They operate in either normal mode, in which they print character output, or graphics mode, in which they print plotter output in the form of small dots. In graphics mode, they generate 10-inch square plots having 600 x 480 visible points. These terminals have no graphic input facilities. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routines:

The DDR for the DTC300 or GSI300 resides in the file *IG.DTC300, and the DDR for the DTC302 resides in the file *IG.DTC302.

Device Recognition and Setup:

The terminal is automatically recognized by IG if it has an answerback of "DTC300" for the DTC300 or GSI300, or "DTC302" for the DTC302, or if the terminal type has been set by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). For graphics mode to work properly, the pitch switch on the terminal must be in the 10-cpi position, not the 12-cpi position, and the parity switch must be set to NONE.

Graphic Output:

Each call to IGDRON regenerates the entire plot.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','RESET')

Resets the terminal from graphics mode to normal mode (e.g., following an attention interrupt).

December 1980

Digital Equipment Corporation GT40 Terminal

General Features:

The DEC GT40 is a vector-refresh-type terminal, based on a PDP-11 minicomputer. It has a rectangular screen with 1024 x 780 visible points. The GT40 has a light pen for picture picking and a tracking cross for graphic coordinate input.

Device-Dependent Routine:

The DDR for the GT40 resides in the file *IG.GT40.

Device Recognition and Setup:

A GT40 is automatically recognized if the GT40 operating system is loaded and running in the GT40. To load this system, the user must start the GT40 at location 166000 (octal) and issue the MTS command:

```
$COPY GRAF:GT40.SYST *SINK*@BIN
```

Graphic Output:

The IGDRON subroutine only deletes and replaces those lines and text which have been changed since the previous call to IGDRON('TERMINAL').

Graphic Input:

The IGXYIN subroutine places a tracking cross on the screen. The user may reposition the cross by pointing to it with the light pen and dragging it to the desired input coordinates. The screen intensity must be greater than 2/3 of full intensity for the tracking cross to work smoothly. The cross position may be returned by hitting any keyboard key. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine places the message "LP" at the lower-right corner of the screen. The user may then point the light pen at the desired subpicture. Upon contact, IGPIKS removes the "LP" message and returns to the calling program.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','CLEAR')

Clears the screen. The next call to IGDRON('TERMINAL') regenerates the screen image. Note: This control operation is similar to the 'ERASE' operation for other terminals.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

Hewlett-Packard Plotters

HP7220A
HP7220C
HP7220S
HP7220T
HP7225A
HP7470
HP7475

General Features:

The Hewlett-Packard 7220A, 7220C, 7220S, 7220T, 7225A, 7470, and 7475 are small, remote plotters designed to be used with many types of terminals. These plotters must be connected between the terminal and the modem. This plotter/terminal combination operates either in normal mode, in which data is passed through the plotter to or from the terminal, or in graphics mode, in which the data is read or written by the plotter. In graphics mode, the 7220A, 7220C, 7220S, and 7220T generate 10-inch square plots having 16000 x 11400 visible points; the 7225A generates 7.5-inch square plots having 11420 x 8140 visible points; the 7470 generates 7.5-inch square plots having 10300 x 7650 visible points; and the 7475 generates either 10-inch square plots having 16640 x 10365 visible points, or 7.5-inch square plots having 10365 x 7962 visible points. These plotters can use 8 different colors: black, red, blue, green, yellow, magenta, cyan, and white. The 7220C and 7220T can hold 8 pens at one time; the 7475 can hold 6 pens; the 7220A and 7220S can hold 4 pens; the 7470 can hold 2 pens; and the 7225A can hold 1 pen. The plotter pen may be moved manually by pressing the "arrow" buttons on the front panel of the plotter. The pen position is used for both graphic coordinate input and picture picking.

Device-Dependant Routine:

The DDR for these plotters resides in the file *IG.HPGL.

Device Recognition and Setup:

These Hewlett-Packard plotters are not automatically recognized by IG. The user must explicitly load the HPGL DDR at execution time by concatenating *IG+*IG.HPGL to the object file.

Graphic Output:

The IGDRON subroutine has two modes of operation. In plotter mode, IGDRON generates a new plot every time it is called. In add-to mode, IGDRON bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the plot. If lines have been moved or deleted, a new plot will be generated. If the X- or Y-margin is not specified by calling IGCTRL (see below) before each call to IGDRON, these margins will be reset to their default values of zero before adding the new information.

The default mode is plotter mode. Upon all calls to IGDRON, the user will be given a message of the following form for each of the plotter's pens:

Load position [number] with [color] pen,

followed by the message:

and press return when you are ready

Upon any call IGDRON that requires the generation of a new plot, the following message will precede the previous messages:

IGDRON -- change paper and mount pens

After mounting the pens (and possibly replacing the paper), the user should press RETURN to begin the generation of the plot.

Graphic Output Attributes:

The following pen numbers are available and may be selected for a given subplot by calling IGATTS:

1 black
2 red
3 blue
4 green
5 yellow
6 magenta
7 cyan
8 white

Graphic Input:

The IGXYIN subroutine makes an implicit call to IGDRON, selects pen #1 (for ease of pointing), turns on the ENTER

light on the front panel of the plotter (if there is one), and waits for user input. The user should: (1) position the pen to the desired coordinates, and (2) press a keyboard key to be used as the termination code for this IGXYIN operation. Pen #1 is put back and the ENTER light is turned off.

The IGPIKS subroutine also makes an implicit call to IGDRON, selects pen #1 (for ease of pointing), turns on the ENTER light (if there is one), and waits for user input. The user should: (1) position the pen on the desired subpicture, and (2) press a keyboard key. Pen #1 is put back and the ENTER light is turned off. If a subpicture was not hit, IGPIKS will select pen #1 and turns the ENTER light back on. The user should repeat steps (1) and (2).

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for these devices and will be ignored.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','SCREEN','FULL')

By default, the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. Thus, the left and right edges of the screen remain unused. This call to IGCTRL redefines the screen coordinates so that the X coordinate ranges from -1 at the left edge to +1 at the right edge. However, the Y coordinate will now be clipped at the top and bottom.

IGCTRL('TERMINAL','SCREEN','SQUARE')

This restores the screen coordinates to their default definitions, in which the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN', 'FULL'). This is the default.

IGCTRL('TERMINAL','SIZE',S1[,S2])

This sets the plot size to S1xS2 inches (or S1xS1, if S2 is not specified). If the requested size exceeds the plotter's maximum, the plot will be clipped. The default value is 10 inches for the 7220A, 7220C, 7220S, and the 7220T; and 7.5 inches for the 7225A, and the 7470. For the 7475, the default depends on the size of the plotter paper.

IGCTRL('TERMINAL','XMAR',XMARGN)

This sets the X margin of the plot to XMARGN. If the resulting X margins exceed the plotter's maximum, the plot will be clipped. The default value is 0.0 inches.

IGCTRL('TERMINAL','YMAR',YMARGN)

This sets the Y margin of the plot to XMARGN. If the resulting Y margins exceed the plotter's maximum, the plot will be clipped. The default value is 0.0 inches.

IGCTRL('TERMINAL','VELOCITY',VELO)

This sets the speed of the pen to VELO cm/sec. VELO must be in the range 1-36. The default speed is 36.

IGCTRL('TERMINAL','PLMODE','PLOTTER')

This specifies that IGDRON is to operate in plotter mode, i.e., that each call to IGDRON is to generate a new plot. This is the default.

IGCTRL('TERMINAL','PLMODE','ADDTO')

This specifies that IGDRON is to operate in add-to mode, i.e., that IGDRON will, if possible, only add lines to the existing plot, but will, if necessary, generate a new plot.

IGCTRL('TERMINAL','PLMODE','ASKVERBOSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given a verbose message explaining plotter and add-to modes and asking which one to use.

IGCTRL('TERMINAL','PLMODE','ASKTERSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user

| will be given a terse message asking which mode to use:
|
| Enter "PLOTTER" or "ADDTO":

Page Revised September 1984 December 1980

Hewlett-Packard 7203A Plotter

General Features:

The Hewlett-Packard 7203A is a small, remote plotter, designed to be used with many different types of terminals. The HP7203A must be connected between the terminal and the modem. This plotter/terminal combination operates in either normal mode, in which data is passed through the plotter to/from the terminal, or graphics mode, in which data is read/written by the plotter. In graphics mode, the HP7203A generates 10-inch square plots having 2500 x 2500 visible points. Each plot is centered in a 15x10-inch sheet of plotter paper. The HP7203A has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the Hewlett-Packard 7203A resides in the file *IG.HP7203.

Device Recognition and Setup:

An HP7203A is not automatically recognized by IG. The user must explicitly load the HP7203A DDR at $RUN time by concatenating *IG+*IG.HP7203 to the object file.

The terminal should be placed in half-duplex mode.

An HP7203A should not be used with a terminal that has either lowercase letters or an answerback.

Graphic Output:

The IGDRON subroutine has two modes of operation. In plotter mode, IGDRON generates a new plot every time it is called. In addto mode, IGDRON bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the plot. If lines have been moved or deleted, a new plot will be generated.

Upon the first call to IGDRON, if the mode has not been specified by calling IGCTRL (see below), the user will be asked which mode to use. Upon any call to IGDRON that requires the generation of a new plot, the user will be given the following message:

IGDRON -- change paper and type return when ready

After replacing the plotter paper, the user should press RETURN to begin the generation of the plot.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','PLMODE','PLOTTER')

Specifies that IGDRON is to operate in plotter mode, i.e., that each call to IGDRON is to generate a new plot.

IGCTRL('TERMINAL','PLMODE','ADDTO')

Specifies that IGDRON is to operate in addto mode, i.e., that IGDRON will if possible only add lines to the existing plot, but will if necessary generate a new plot.

IGCTRL('TERMINAL','PLMODE','ASKVERBOSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given a verbose message explaining plotter and addto modes and asking which one to use.

IGCTRL('TERMINAL','PLMODE','ASKTERSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given the following terse message asking which mode to use:

ENTER "PLOTTER" OR "ADDTO"

IGCTRL('TERMINAL','RESET')

Resets the plotter/terminal from graphics mode to normal mode (e.g., following an attention interrupt).

Hewlett-Packard 7221A Plotter

General Features:

The Hewlett-Packard 7221A is a small, remote plotter, designed to be used with many different types of terminals. The HP7221A must be connected between the terminal and the modem. This plotter/terminal combination operates in either normal mode, in which data is passed through the plotter to/from the terminal, or graphics mode, in which data is read/written by the plotter. In graphics mode, the HP7221A generates 15.75x 11-inch plots having 3040 x 2000 visible points. The plotter pen may be moved manually by pressing the "arrow" buttons on the front panel of the plotter. The pen position is used for both graphic coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the HP7221A resides in the file *IG.HP7221.

Device Recognition and Setup:

An HP7221A is not automatically recognized by IG. The user must explicitly load the HP7221A DDR at $RUN time by concatenating *IG+*IG.HP7221 to the object file.

Graphic Output:

The IGDRON subroutine has two modes of operation. In plotter mode, IGDRON generates a new plot every time it is called. In addto mode, IGDRON bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the plot. If lines have been moved or deleted, a new plot will be generated.

Upon the first call to IGDRON, if the mode has not been specified by calling IGCTRL (see below), the user will be asked which mode to use. Upon any call to IGDRON that requires the generation of a new plot, the user will be given the following message:

```
IGDRON -- change paper and type return when ready
```

After replacing the plotter paper, the user should press RETURN to begin the generation of the plot.

Graphic Input:

The IGXYIN subroutine makes an implicit call to IGDRON, turns on the ENTER light on the front panel of the plotter, and waits for user input. The user should then: (1) position the pen to the desired coordinates, (2) press a keyboard key to be used as the termination code for this IGXYIN operation, and (3) press the ENTER button and immediately lift his/her hand.

The IGPIKS subroutine also makes an implicit call to IGDRON, turns on the ENTER light, and waits for user input. The user should then: (1) position the pen on the desired subpicture, (2) press a keyboard key, and (3) press the ENTER button and immediately lift his/her hand. If a subpicture was not hit, IGPIKS will turn the ENTER light back on. The user should then move the pen closer to the subpicture and repeat steps (2) and (3).

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','PLMODE','PLOTTER')

Specifies that IGDRON is to operate in plotter mode, i.e., that each call to IGDRON is to generate a new plot.

IGCTRL('TERMINAL','PLMODE','ADDTO')

Specifies that IGDRON is to operate in addto mode, i.e., that IGDRON will if possible only add lines to the existing plot, but will if necessary generate a new plot.

IGCTRL('TERMINAL','PLMODE','ASKVERBOSE')

If the IGDRON mode of operation has not been specified by the time of the <u>first</u> IGDRON call, the user will be given a verbose message explaining plotter and addto modes and asking which one to use.

IGCTRL('TERMINAL','PLMODE','ASKTERSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given the following terse message asking which mode to use:

ENTER "PLOTTER" OR "ADDTO"

IGCTRL('TERMINAL','SCREEN','FULL')

By default, the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. Thus, the left and right edges of the screen remain unused. This call to IGCTRL redefines the screen coordinates so that the X coordinate ranges from -1 at the left edge to +1 at the right edge. However, the Y coordinate will now be clipped at the top and bottom.

IGCTRL('TERMINAL','SCREEN','SQUARE')

Restores the screen coordinates to their default definitions, in which the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','FULL').

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','RESET')

Resets the plotter/terminal from graphics mode to normal mode (e.g., following an attention interrupt).

December 1980

Houston Instruments Data Plotter 11

General Features:

The Houston Instruments Data Plotter 11 is a small, remote plotter, designed to be used with many different types of terminals. The HIDP11 must be connected between the terminal and the modem. This plotter/terminal combination operates in either normal mode, in which data is passed through the plotter to/from the terminal, or graphics mode, in which data is read/written by the plotter. In graphics mode, the HIDP11 generates 11-inch square plots having 301 x 301 visible points. The HIDP11 has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the Houston Instruments Data Plotter 11 resides in the file *IG.HIDP11.

Device Recognition and Setup:

A Houston Instruments Data Plotter 11 is not automatically recognized by IG. The user must explicitly load the HIDP11 DDR at $RUN time by concatenating *IG+*IG.HIDP11 to the object file.

Graphic Output:

The IGDRON subroutine has two modes of operation. In plotter mode, IGDRON generates a new plot every time it is called. In addto mode, IGDRON bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the plot. If lines have been moved or deleted, a new plot will be generated.

Upon the first call to IGDRON, if the mode has not been specified by calling IGCTRL (see below), the user will be asked which mode to use. Upon any call to IGDRON that requires the generation of a new plot, the user will be given the following message:

```
IGDRON -- change paper and type return when ready
```

After replacing the plotter paper, the user should press RETURN to begin the generation of the plot.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','PLMODE','PLOTTER')

Specifies that IGDRON is to operate in plotter mode, i.e., that each call to IGDRON is to generate a new plot.

IGCTRL('TERMINAL','PLMODE','ADDTO')

Specifies that IGDRON is to operate in addto mode, i.e., that IGDRON will if possible only add lines to the existing plot, but will if necessary generate a new plot.

IGCTRL('TERMINAL','PLMODE','ASKVERBOSE')

If the IGDRON mode of operation has not been specified by the time of the <u>first</u> IGDRON call, the user will be given a verbose message explaining plotter and addto modes and asking which one to use.

IGCTRL('TERMINAL','PLMODE','ASKTERSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given the following terse message asking which mode to use:

ENTER "PLOTTER" OR "ADDTO"

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

Hughes Conographic C-9 Terminal

General Features:

The Hughes Conographic C-9 is a raster-refresh-type terminal with a hardware vector generator and a selective-erase capability. It has a rectangular screen with 2048 x 1536 visible points. The Hughes has a cursor that can be moved about the screen by manipulating a joystick. This cursor is used for both graphic coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the Hughes Conographic C-9 resides in the file *IG.HUGHES.

Device Recognition and Setup:

A Hughes Conographic C-9 is automatically recognized if the terminal type has been set to "HUGHES" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS).

Graphic Output:

The IGDRON subroutine takes advantage of the selective-erase capability of the Hughes to speed up small modifications to the screen image. When IGDRON is called, it determines how many data bytes are necessary either (1) to selectively erase the deleted lines and display the added lines, or (2) to erase the entire screen and regenerate the entire image (as with storage-tube-type terminals). IGDRON then picks the quickest method.

Graphic Input:

The IGXYIN subroutine waits for the user to move the cursor to the desired position on the display screen. This may be done by manipulating the joystick. The cursor position may be returned by hitting any keyboard key. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the cursor. The user should position it on the desired subpicture and then press a keyboard key. If a subpicture was not hit, IGPIKS will

immediately reactivate the cursor. The user should move it closer to the desired subpicture and again press a keyboard key.

Control Operations:

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IBM-3270-Compatible Terminals

Courier C-270
IBM 3278

General Features:

The IBM-3270-compatible terminals consist of a storage-tube-type screen with a separate keyboard unit. These terminals are designed primarily for nongraphic, keyboard I/O operations. When they are used with IG, graphic output is simulated by a full screen display containing 80 characters by 24, 35, or 43 lines (the exact number depending on the model of the terminal). Graphic input is performed by positioning the keyboard cursor within this display.

Device-Dependent Routine:

The DDR for the IBM-3270-compatible terminals resides in the file *IG.3270.

Device Recognition and Setup:

The IBM-3270-compatible terminals are recognized automatically by IG.

Graphic Output:

Each call to IGDRON regenerates the entire display. At each point, a character is chosen to approximate the shape of the lines passing through that point.

Graphic Input:

The IGXYIN subroutine writes the footer "GRAPHIC INPUT" in the lower right corner of the screen. The user may then specify the input coordinates by positioning the keyboard cursor. The coordinates may be returned by pressing one of the program function keys or the ENTER key. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also writes the footer "GRAPHIC INPUT" in the lower-right corner of the screen. The user may move the keyboard cursor to pick the desired subpicture, then press one of the program function keys or the ENTER key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge of the display area to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the display area.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge of the display area to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

Magnavox 12000 Terminal

General Features:

The Magnavox 12000 is a plasma-panel terminal with a selective-erase capability. It has an 8.5-inch square screen with 512 x 512 visible points. The MX12000 has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the MX12000 resides in the file *IG.MX12000.

Device Recognition and Setup:

An MX12000 is not automatically recognized by IG. The user must explicitly load the MX12000 DDR at $RUN time by concatenating *IG+*IG.MX12000 to the object file.

Graphic Output:

The IGDRON subroutine takes advantage of the selective-erase capability of the MX12000 to speed up small modifications to the screen image. When IGDRON is called, it determines how many data bytes are necessary either (1) to selectively erase the deleted lines and display the added lines, or (2) to erase the entire screen and regenerate the entire image (as with storage-tube-type terminals). IGDRON then picks the quickest method.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a

portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

Nongraphics Terminals

Teletypes
DECwriters
IBM-2741-Compatible Terminals
etc.

General Features:

Nongraphics terminals have no graphic I/O functions. When they are used with IG, graphic output operations are replaced by printed descriptions of the IG data structure. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for nongraphics terminals resides in the file *IG.TTY.

Device Recognition and Setup:

Nongraphics terminals are recognized automatically unless the user is signed on at a special 300-baud terminal or is using an RFL dataset. In the latter cases, the user must either explicitly load the nongraphics terminals DDR at $RUN time by concatenating *IG+*IG.TTY to the object file or indicate to the system that he or she is on a Teletype-compatible device. For further details, see MTS Volume 4, Terminals and Networks in MTS.

Graphic Output:

Instead of generating actual graphic output, a call to IGDRON('TERMINAL') generates the message

```
CALL IGDRON('TERM')
```

and then generates a text description of the changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). On the first call to IGDRON('TERMINAL') the user is prompted with the question

```
WHERE TO DUMP COORDINATES?
```

for an FDname on which to write the text description. On all subsequent calls, the text description is written to that FDname. The first line of the text description gives an

operation type (CREATE, DELETE, ADD TO, or REDRAW), followed by the name of the altered subpicture, followed by a list of

December 1980

the names of the higher-level subpictures that trace the connection back to '*MP*'. Subsequent lines of the text description give the coordinates of the lines in the altered subpicture, transformed to -1 to +1 screen coordinates, and the text in the subpicture, printed as text. Note that this information is normally used to generate the screen image for a graphics device. Thus, all zero length or colinear line segments are deleted. In the present case, however, the clipping of lines at the edges of the "screen" is bypassed. Thus, numbers may be printed outside the range -1 to +1. This is useful for finding "where the picture went" if it is not properly scaled or positioned within the screen boundaries.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

Any calls to IGCTRL('TERMINAL',...) will cause the parameter list to be printed on the terminal for error-checking purposes.

Object-Saving Files or Devices

General Features:

Any file or storage device can be used to save an object, representing the current data structure, in such a way that it can later be reloaded (see the section "Saving the Current Data Structure as an Object").

Device-Dependent Routine:

The DDR resides in the file *IG.SAVE.

Device Recognition and Setup:

The DDR is always accessible as the 'SAVE' device.

Graphic Output:

A call to IGDRON('SAVE') dumps the object representing the current data structure onto logical I/O unit SPUNCH or a file or device specified by a call to IGCTRL('SAVE','OUTPUT', 'fdname ') This object may later be reloaded by calling the IGLOAD subroutine.

Graphic Input:

Graphic input operations are not performed with the 'SAVE' device.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for the 'SAVE' device and will be ignored.

IGCTRL('SAVE','OUTPUT','fdname ')

Specifies "fdname" as the destination for the next object to be dumped by calling IGDRON('SAVE'). Note that "fdname" must be followed by a trailing blank.

IGCTRL('SAVE','NAME','objnam')

Specifies "objnam" as the name for the next object to be saved. The first time a name is not specified, the default name '0000' (four zeros) will be assigned. Subsequently, every time a name is not specified, the default name will be incremented by 1 and assigned.

Princeton Electronics Products 801 Terminal

General Features:

The Princeton Electronics Products 801 is a lithicon storage-tube-type terminal with a selective-erase capability. It has a 10-inch square screen with 4096 x 4096 visible points. The PEP 801 has a cursor that can be moved about the screen by manipulating the joystick to the right of the keyboard. This cursor is used for both graphic coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the PEP 801 resides in the file *IG.PEP.

Device Recognition and Setup:

A PEP 801 is automatically recognized by IG if the terminal type has been set to "PEP801" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). Otherwise, the user must explicitly load the PEP 801 DDR at $RUN time by concatenating *IG+*IG.PEP to the object file.

Graphic Output:

The IGDRON subroutine takes advantage of the selective-erase capability of the PEP 801 to speed up small modifications to the screen image. When IGDRON is called, it determines how many data bytes are necessary either (1) to selectively erase the deleted lines and display the added lines, or (2) to erase the entire screen and regenerate the entire image (as with storage-tube-type terminals). IGDRON then picks the quickest method.

Graphic Input:

The IGXYIN subroutine waits for the user to move the cursor to the desired position on the display screen. This may be done by manipulating the joystick to the right of the keyboard. The cursor position may be returned by hitting any keyboard key. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the cursor. The user should position it on the desired subpicture and then press a keyboard key. If a subpicture was not hit, IGPIKS will immediately reactivate the cursor. The user should move it closer to the desired subpicture and again press a keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

Printer-Plot Devices

Line Printers
Terminals

General Features:

The current *MSINK* device may be used for graphic output, in the form of one-page plots containing 100 characters x 60 lines.

Device-Dependent Routine:

The printer-plot DDR resides in the file *IG.PRNT.

Device Recognition and Setup:

To use the printer-plot DDR for the 'TERMINAL' device, the user must explicitly load this DDR at $RUN time by concatenating *IG+*IG.PRNT to the object file. The printer-plot DDR is primarily useful in batch mode.

Graphic Output:

Each call to IGDRON generates a one-page plot containing 100 characters x 60 lines. At each point in the plot, a character is chosen to approximate the shape of the lines passing through that point.

As each plot is generated, it is automatically written on *MSINK*.

Graphic Input:

Graphic input operations are illegal for this device type, and any calls to IGXYIN or IGPIKS will cause errors.

Control Operations:

No IGCTRL parameters are recognized for this device.

December 1980

Qume Sprint 5 Terminal with Plot Option

General Features:

The Qume Sprint 5 is a hard-copy terminal with a Diablo-type print mechanism. It operates in either normal mode, in which it prints character output, or graphics mode, in which it prints plotter output in the form of small dots. In graphics mode, it generates 10-inch square plots having 600 x 480 visible points. The Qume has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the Qume resides in the file *IG.QUMES.

Device Recognition and Setup:

A Qume is not automatically recognized by IG. The user must explicitly load the Qume DDR at $RUN time by concatenating *IG+*IG.QUMES to the object file.

Graphic Output:

Each call to IGDRON regenerates the entire plot.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','RESET')

Resets the terminal from graphics mode to normal mode (e.g., following an attention interrupt).

December 1980

Ramtek 6200A Terminal

General Features:

The Ramtek 6200A is a color raster-refresh device with a hardware vector generator. It has a rectangular screen with an aspect ratio of 0.75 and with 512 x 256 visible points. Thus, the resolution in the X dimension is finer than that in the Y dimension. For graphic input, the Ramtek 6200A has crosshairs that can be moved by pressing the "arrow" buttons.

Device-Dependent Routine:

The DDR for the Ramtek 6200A resides in the file *IG.RK6200.

Device Recognition and Setup:

An RK6200A is not automatically recognized by IG. The user must explicitly load the RK6200A DDR at $RUN time by concatenating *IG+*IG.RK6200 to the object file.

Graphic Output:

The IGDRON subroutine only deletes and replaces those lines and text which have been changed since the previous call to IGDRON('TERMINAL').

Graphic Output Attributes:

Currently, the relationship between pen numbers and colors is fixed as follows:

1	black
2	red
3	blue
4	green

However, the RK6200A actually has 24 colors (only eight of which can be used in any one screen image). In the future, the user will be able to call IGCTRL to redefine the mapping of pen numbers into colors. Thus, all 24 colors will be available (though not all at the same time).

Graphic Input:

The IGXYIN subroutine activates a pair of crosshairs on the screen. These may be moved around by pressing the "arrow" buttons. Any keyboard key may be pressed to return the position of the crosshairs. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the crosshairs. The user should position the crosshairs on the desired subpicture and then press a keyboard key. If a subpicture was not hit, IGPIKS will immediately reactivate the crosshairs. The user should move them closer to the desired subpicture and again press a keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

Trendata 4000 Terminal with Plot Option

General Features:

The Trendata 4000 is a hard-copy terminal with a Diablo-type print mechanism. It operates in either normal mode, in which it prints character output, or graphics mode, in which it prints plotter output in the form of small dots. In graphics mode, it generates 10-inch square plots having 600 x 480 visible points. The TD4000 has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the TD4000 resides in the file *IG.TD4000.

Device Recognition and Setup:

A TD4000 is automatically recognized by IG if it has an answerback of "TD4000" or the terminal type has been set to "TD4000" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). The user may explicitly load the TD4000 DDR at $RUN time by concatenating *IG+*IG.TD4000 to the object file. The parity switch should be set to NONE or plotting will not work properly.

Graphic Output:

Each call to IGDRON regenerates the entire plot.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt

for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

No IGCTRL parameters are recognized for this device.

Tektronix 4002 Terminal with Graphic Input Options

General Features:

The Tektronix 4002 is a storage-tube-type terminal. It has a rectangular screen with 1024 x 760 visible points. It operates in either normal mode, in which it reads keyboard input and writes character output, or graphics mode, in which it does graphic I/O. When fitted with a 4901 Interactive Graphic Unit and a joystick, the TX4002 is capable of graphic input. The joystick can be used to move a pair of crosshairs around the screen. These crosshairs are used for both graphic coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the TX4002 resides in the file *IG.TX4002.

Device Recognition and Setup:

A TX4002 is automatically recognized if it is connected to the UMnet Computer Network at 300 baud or through an RFL data set or Vadic data set. A TX4002 is automatically recognized if the terminal type has been set to "T4002" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). The KEYB/AUX selector switch above the keyboard must be in the KEYB and AUX position (both green and white lamps lit) and the joystick must be turned on (thumbwheel switch on the base of the joystick).

Graphic Output:

The IGDRON subroutine bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the screen image. If lines have been moved or deleted, the screen will be erased and the entire image regenerated.

Graphic Input:

The IGXYIN subroutine activates a pair of crosshairs on the screen. These may be moved around by using the joystick. Any keyboard key may be pressed to return the position of the crosshairs. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the crosshairs. The user should position the crosshairs on the desired subpicture and then press a keyboard key. If a subpicture was not hit, IGPIKS will immediately reactivate the crosshairs. The user should move them closer to the desired subpicture and again press a keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','RESET')

Resets the terminal from graphics mode to normal mode (e.g., following an attention interrupt).

December 1980

Tektronix 4006 Terminal

General Features:

The Tektronix 4006 is a storage-tube-type terminal. It has a rectangular screen with 1024 x 781 visible points. The TX4006 has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the TX4006 resides in the file *IG.TX4006.

Device Recognition and Setup:

A TX4006 is automatically recognized if the terminal type has been set to "T4006" by giving the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). When using this type of terminal with the UMnet Computer Network, it is not possible or advisable to queue input text ahead of the program during device recognition or before any IGDRON call that does not erase the screen. This input queueing interferes with the implicit cursor-read operation performed at these times. Also note that input text (other than CTRL-E) is ignored while the screen image is being generated.

Graphic Output:

The IGDRON subroutine bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the screen image. If lines have been moved or deleted, the screen will be erased and the entire image regenerated.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON ('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','FULL')

By default, the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. Thus, the left and right edges of the screen remain unused. This call to IGCTRL redefines the screen coordinates so that the X coordinate ranges from -1 at the left edge to +1 at the right edge. However, the Y coordinate will now be clipped at the top and bottom.

IGCTRL('TERMINAL','SCREEN','SQUARE')

Restores the screen coordinates to their default definitions, in which the -1 to +1 screen coordinates occupy

the largest possible square centered within the screen. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','FULL').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

Page Revised September 1984 December 1980

Tektronix 4010 Series Terminals

Tektronix 4010
Tektronix 4012
Tektronix 4014

General Features:

The Tektronix 4010 Series are storage-tube-type terminals. The TX4010 and TX4012 have rectangular screens with 1024 x 781 visible points. The TX4014 has a rectangular screen with 4096 x 3120 visible points. For graphic input, these terminals have crosshairs that can be moved by rotating the thumbwheels to the right of the keyboard. The crosshairs are used for both coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the TX4010 and TX4012 resides in the file *IG.TX4010, and the DDR for the TX4014 resides in the file *IG.TX4014.

Device Recognition and Setup:

A Tektronix 4010 Series terminal is automatically recognized if it is connected through the UMnet Computer Network at 300 baud or through an RFL or Vadic dataset. A Tektronix 4010 Series terminal is automatically recognized if the terminal type has been set by the TERMINAL device command to the appropriate terminal control unit (see MTS Volume 4, Terminals and Networks in MTS). The strappable option for "Graphic Input Terminators" must be set to "CR only". When using this type of terminal with the UMnet Computer Network, it is not possible or advisable to queue input text ahead of the program during device recognition or before any IGDRON call that does not erase the screen. This input queueing interferes with the implicit cursor-read operation performed at these times. Also note that input text (other than CTRL-E) is ignored while the screen image is being generated.

Graphic Output:

The IGDRON subroutine bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the screen image. If lines have been moved or deleted, the screen will be erased and the entire image regenerated.

Graphic Input:

The IGXYIN subroutine activates a pair of crosshairs on the screen. These may be moved about by rotating the thumbwheels to the right of the keyboard. Any keyboard key may be pressed to return the position of the crosshairs. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the crosshairs. The user should position the crosshairs on the desired subpicture and then press a keyboard key. If a subpicture was not hit, IGPIKS will immediately reactivate the crosshairs. The user should move them closer to the desired subpicture and again press a keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','4953','SPPM')

Enables the use of the optional 4953 Graphics Tablet, and places it in single-point-presence mode. See the Tektronix 4953 Graphics Tablet Instruction Manual.

IGCTRL('TERMINAL','4953','SPPF')

Enables the use of the optional 4953 Graphics Tablet, and places it in single-point-presence-full mode. See the Tektronix 4953 Graphics Tablet Instruction Manual.

IGCTRL('TERMINAL','4953','OFF ')

Disables the use of the 4953 Graphics Tablet. This is the default mode.

December 1980

Tektronix 4025 or 4027 Terminals

TX4025
TX4027

General Features:

The Tektronix 4025 and 4027 are raster-refresh devices with hardware vector generators. They have rectangular screens with 568 x 434 visible points. The TX4025 is monochromatic while the TX4027 has colors. For graphic input, these terminals have crosshairs that can be moved by pressing the "arrow" buttons at the right of the keyboard.

Device-Dependent Routine:

The DDR for the TX4025 resides in the file *IG.TX4025, and the DDR for the TX4027 resides in the file *IG.TX4027.

Device Recognition and Setup:

A TX4025 or TX4027 is automatically recognized by IG if the terminal type has been set to "T4025" or "T4027" by giving the TERMINAL device command to the terminal control unit (see MTS Volume 4, Terminals and Networks in MTS).

Graphic Output:

The IGDRON subroutine bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the screen image. If lines have been moved or deleted, the screen will be erased and the entire image regenerated.

Graphic Output Attributes:

Currently, the relationship between pen numbers and colors is fixed as follows (for the TX4027 only):

1	black
2	red
3	blue
4	green

However, the TX4027 actually has 32 colors (only eight of which can be used in any one screen image). In the future, the user will be able to call IGCTRL to redefine the mapping of pen numbers into colors. Thus, all 32 colors will be available (though not all at the same time).

Graphic Input:

The IGXYIN subroutine activates a pair of crosshairs on the screen. These may be moved around by pressing the "arrow" buttons at the right of the keyboard. Any keyboard key may be pressed to return the position of the crosshairs. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the crosshairs. The user should position the crosshairs on the desired subpicture and then press a keyboard key. If a subpicture was not hit, IGPIKS will immediately reactivate the crosshairs. The user should move them closer to the desired subpicture and again press a keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

Erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the

top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

Restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

Positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

Tektronix 4100 Series Terminals

Tektronix 4105
Tektronix 4107
Tektronix 4109
Tektronix 4113

General Features:

The Tektronix 4100 series are color raster terminals. The 4105 has 480 x 360 resolution on a 13-inch screen, the 4107 has 640 x 480 resolution on a 13-inch screen, and the 4109 and 4113 have 640 x 480 resolution on a 19-inch screen.

For graphic input the Tektronix 4105, 4107, and 4109 have crosshairs that can be moved by using the joydisk located on the upper left of the keyboard. The 4113 has crosshairs that can be moved by rotating the thumbwheels located to the right of the keyboard.

Device-Dependent Routine:

The DDR for the Tektronix 4100 series resides in the file *IG.TX4113.

Device Recognition and Setup:

A Tektronix 4100 series terminal is automatically recognized by IG if the terminal type has been set currently to either "T4105" or "T4113" (see MTS Volume 4, Terminals and Networks in MTS). This device driver may also be explicitly loaded at execution time by concatenating *IG+*IG.TX4113 to the object file.

For the Tektronix 4100 series terminals to work with IG, the End-Of-Message (EOM) Characters must be set to CR and LF, and the Bypass-Cancel Character must be set to NU.

Graphic Output:

The IGDRON subroutine bases its actions on the changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the screen image. If lines have been moved or deleted, currently the screen will be erased and the entire image regenerated.

When IG is first initialized, the dialog area is set up so that it is enabled but left invisible. Should any communication need to be done to the user, it is the responsibility of

the application program to make the dialog area visible. It is also the responsibility of the application program to make the dialog area invisible again where necessary. This can be done with a call to the IGCTRL routine described below.

Alternatively, the control of the visibility of the dialog area can be done through the use of the Dialog key on the keyboard.

Graphic Output Attributes:

The 4105 can display eight colors at one time, the 4107 and the 4109 can display sixteen colors at one time, and the 4113 can display either eight or sixteen colors at one time depending on the terminal. For these terminals, the IG pen numbers 1 - 8 represent the eight colors as follows:

1 white
2 red
3 blue
4 green
5 yellow
6 magenta
7 cyan
8 black

On the terminals in this series which have more than eight colors, the IG pen numbers 9 and above will be the default colors of the terminal.

The correspondence of the colors to a specific color index may be redefined (see the IGCTRL section below for details).

The 4105, 4107, 4109, and 4113 terminals all support hardware polygons and fills. The polygon edge pen numbers all correspond to the normal pen numbers, and if any pen number has had its color definition changed, then the corresponding edge pen number will be changed also.

The polygon fill numbers are set up so that the first sixteen fill numbers (0-15) correspond to the sixteen pen colors. Fill numbers above fifteen are for patterns available by the terminal.

The scheme for mapping the PFILL numbers to the actual Tektronix fill patterns is:

IG Numbers		Tektronix Numbers	
0 to 15	[remove the sign]	-15 to 0	(solid colors)
16 to 31	[number+15]	1 to 16	(predefined)
---		17 to 49	** invalid
32 to 156	[number-18]	50 to 174	(dither patterns)
> 157	** invalid	>174	** invalid

Graphic Input:

The IGXYIN subroutine activates a pair of crosshairs on the screen. These may be moved around by pressing on the edges of the joydisk or by rotating the thumbwheels depending on which terminal being used. Any keyboard key may be pressed to return the position of the crosshairs. The termination code (the IGXYIN function value) is given in Table 1 in the section "Graphic Input".

The IGPIKS subroutine also activates the crosshairs. The user should position the crosshairs on the desired subpicture and then then press a keyboard key. If a subpicture was not hit, IGPIKS will immediately reactivate the crosshairs. The user should move them closer to the desired subpicture and again press a keyboard key.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','ERASE')

This erases the screen. The next call to IGDRON('TERMINAL') regenerates the screen image.

IGCTRL('TERMINAL','KEEP',ISW)

If ISW=1, this enables keep mode. When IGDRON is called, lines are only added to the screen image. Lines that would normally be deleted are allowed to remain on the screen. The screen may be explicitly erased by making a call to IGCTRL('TERMINAL','ERASE').

If ISW=0, this disables keep mode and reenables automatic-erase mode (the default). When IGDRON has to delete lines from the screen image, it will automatically erase the screen and regenerate the entire image.

IGCTRL('TERMINAL','SCREEN','SQUARE')

By default, the screen coordinates are defined so that the X coordinate ranges from -1 at the left edge to +1 at

the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL redefines the screen coordinates so that they occupy the largest square centered within the screen.

IGCTRL('TERMINAL','SCREEN','FULL')

This restores the screen coordinates to their default definitions, in which the X coordinate ranges from -1 at the left edge to +1 at the right edge, while the Y coordinate is clipped at the top and bottom. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','SQUARE').

IGCTRL('TERMINAL','POSN',X,Y)

This positions the keyboard cursor to the coordinates (X,Y) in the -1 to +1 screen coordinate system.

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','POLYFILL','SOFTWARE')

This tells IG to use software generated polygons and fills instead of hardware polygons and fills available in the device. IG will use the hardware polygons and fills by default.

IGCTRL('TERMINAL','POLYFILL','HARDWARE')

This tells IG to go back to the default of using hardware polygons and fills instead of using ones generated in software.

IGCTRL('TERMINAL','DAVI',ISW)

If ISW=1, then the dialog area will become visible. If ISW=0, then the dialog area will become invisible.

IGCTRL('TERMINAL','COLT',NVAL,ISTART,IRED,IGREEN,IBLUE)

This redefines the color associated with a specific color index. This is accomplished by specifying the red, green, and blue (RGB) components of a color for one or more color indices.

NVAL
A positive integer specifying the number of color indices that are to be set.

ISTART
An integer value specifying the color index to which to start assigning RGB color components.

IRED, IGREEN, IBLUE
Halfword integer arrays, corresponding elements of these three arrays give the red, green, and blue color components for a particular color index. The integer value of a particular array element can be in the range 0 - 255. This value determines the intensity of a particular color component, with 0 specifying minimum intensity and 255 specifying maximum intensity.

IGCTRL('TERMINAL','ESCAPE',LEN,AREA)

This allows terminal functions that are not controlled by IG to be accessed by sending the appropiate Tektronix 4100 series command.

LEN
Halfword integer specifying the length in bytes of AREA.

AREA
The location of the character string denoting the escape command.

Note that the use of this call means of controlling the terminal can make the application program device-dependent and thus not able to be used on different terminals.

Tektronix 4662 Plotter

General Features:

The Tektronix 4662 is a small, remote plotter, designed to be used with many different types of terminals. The TX4662 must be connected between the terminal and the modem. This plotter/terminal combination operates in either normal mode, in which data is passed through the plotter to/from the terminal, or graphics mode, in which data is read/written by the plotter. In graphics mode, the TX4662 generates rectangular plots having 4096 x 3120 visible points. The plotter pen may be moved manually by means of the joystick on the front panel of the plotter. The pen position is used for both graphic coordinate input and picture picking.

Device-Dependent Routine:

The DDR for the Tektronix 4662 Plotter resides in the file *IG.TX4662.

Device Recognition and Setup:

A TX4662 is not automatically recognized by IG. The user must explicitly load the TX4662 DDR at $RUN time by concatenating *IG+*IG.TX4662 to the object file. The switches on the back of the TX4662 should be set to 2221 for proper use at 300 baud. This causes the plotter to be ON when the power is turned ON. Unless the plotter is turned logically OFF, all input and subsequent output from MTS will be echoed. In order to turn the plotter logically OFF, the user should place the terminal in LOCAL and depress the ESCAPE key, and then the letters capital A and capital F (3 distinct key strokes). The terminal should then be taken out of LOCAL.

Graphic Output:

The IGDRON subroutine has two modes of operation. In plotter mode, IGDRON generates a new plot every time it is called. In addto mode, IGDRON bases its actions on the kinds of changes that have been made to the data structure since the previous call to IGDRON('TERMINAL'). If lines have been added to the data structure, corresponding lines will be added to the plot. If lines have been moved or deleted, a new plot will be generated.

Page Revised September 1984 December 1980

Upon the first call to IGDRON, if the mode has not been specified by calling IGCTRL (see below), the user will be asked which mode to use. Upon any call to IGDRON that requires the generation of a new plot, the user will be given the following message:

IGDRON -- change paper and type return when ready

After replacing the plotter paper, the user should press RETURN to begin the generation of the plot.

Graphic Input:

The IGXYIN subroutine makes an implicit call to IGDRON, turns on the prompt light on the plotter, and waits for user input. The user should then: (1) position the pen to the desired coordinates, (2) press a keyboard key to be used as the termination code for this IGXYIN operation, and (3) press the CALL button and immediately lift his/her hand.

The IGPIKS subroutine also makes an implicit call to IGDRON, turns on the prompt light, and waits for user input. The user should then: (1) position the pen on the desired subpicture, (2) press a keyboard key, and (3) press the CALL button and immediately lift his/her hand. If a subpicture was not hit, IGPIKS will turn the prompt light back on. The user should then move the pen closer to the subpicture and repeat steps (2) and (3).

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','PLMODE','PLOTTER')

Specifies that IGDRON is to operate in plotter mode, i.e., that each call to IGDRON is to generate a new plot.

IGCTRL('TERMINAL','PLMODE','ADDTO')

Specifies that IGDRON is to operate in addto mode, i.e., that IGDRON will if possible only add lines to the existing plot, but will if necessary generate a new plot.

IGCTRL('TERMINAL','PLMODE','ASKVERBOSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given

a verbose message explaining plotter and addto modes and asking which one to use.

IGCTRL('TERMINAL','PLMODE','ASKTERSE')

If the IGDRON mode of operation has not been specified by the time of the first IGDRON call, the user will be given the following terse message asking which mode to use:

ENTER "PLOTTER" OR "ADDTO"

IGCTRL('TERMINAL','SCREEN','FULL')

By default, the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. Thus, the left and right edges of the screen remain unused. This call to IGCTRL redefines the screen coordinates so that the X coordinate ranges from -1 at the left edge to +1 at the right edge. However, the Y coordinate will now be clipped at the top and bottom.

IGCTRL('TERMINAL','SCREEN','SQUARE')

Restores the screen coordinates to their default definitions, in which the -1 to +1 screen coordinates occupy the largest possible square centered within the screen. This call to IGCTRL reverses the effect of a previous call to IGCTRL('TERMINAL','SCREEN','FULL').

IGCTRL('TERMINAL','PICKBOX',S)

A pickbox is used to determine which subpicture is picked during an IGPIKS call. By default, the size of this pickbox is 0.05, expressed as a fraction of the -1 to +1 viewport. If resolution between subpictures becomes a problem, a finer pickbox may be specified by this IGCTRL call. The variable S, which should be in the range from 0 to 1, gives the size of the pickbox.

IGCTRL('TERMINAL','RESET')

Resets the plotter/terminal from graphics mode to normal mode (e.g., following an attention interrupt).

Xerox 1620 Terminal with Plot Option

General Features:

The Xerox 1620 is a hard-copy terminal with a Diablo-type print mechanism. It operates in either normal mode, in which it prints character output, or graphics mode, in which it prints plotter output in the form of small dots. In graphics mode, it generates 10-inch square plots having 600 x 480 visible points. The Xerox 1620 has no graphic input facility. Graphic input operations are simulated by typing in the coordinate values.

Device-Dependent Routine:

The DDR for the Xerox 1620 resides in the file *IG.XX1620.

Device Recognition and Setup:

A Xerox 1620 terminal is automatically recognized by IG if it has an answerback of "X1620" or the terminal type has been set to "X1620" by giving the TERMINAL device command to the appropriate terminal control unit. (see MTS Volume 4, Terminals and Networks in MTS). The parity switch should be set to NONE or plotting will not work properly.

Graphic Output:

Each call to IGDRON regenerates the entire plot.

Graphic Input:

The IGXYIN subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a termination code (an integer value between 0 and 63). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input.

The IGPIKS subroutine prompts the user for X and Y coordinate values (two floating-point values between -1 and +1) and a subpicture name (a four-character string). These must be entered on one input line, separated by commas. Any invalid characters will generate an error comment and another prompt for input. The IG data structure is examined to see if a portion of the named subpicture is near the coordinates

entered. If so, the subpicture is picked and a return is made from IGPIKS to the calling program. If not, the message

NO HIT, TRY AGAIN

is printed and the user is prompted for more input.

Control Operations:

The IGCTRL subroutine has the following valid parameters. Any other parameters are invalid for this device type and will be ignored.

IGCTRL('TERMINAL','RESET')

Resets the terminal from graphics mode to normal mode (e.g., following an attention interrupt).

December 1980

ADDITIONAL DEVICES SUPPORTED BY IG

The following devices are supported by IG but are not currently in use on MTS in Ann Arbor. Users who wish to use these devices are invited to contact the Computing Center for further details.

Device	Type
Adage Graphics Terminal	Refresh
DEC 338 Terminal	Refresh
DEC 339 Terminal	Refresh
GenCom 3000 Terminal	Diablo
Hewlett-Packard 2698 Terminal	Storage-Tube
Printronix 600 Plotter	Plotter
Vucom Terminal	Storage-Tube
Zeta Plotter	Plotter

December 1980

APPENDIX J: INTERFACE WITH THE PLOT DESCRIPTION SYSTEM

The Plot Description System (PDS) is a graphics subroutine package designed especially for producing CalComp plots (see MTS Volume 11, Plot Description System). PDS resides in the public file *PLOTSYS. PDS includes subroutines that generate coordinate axes, grids, graphs, circles, curves, text, etc. The IG system contains an interface with PDS. This has two main purposes:

(1) To allow PDS users to run their programs interactively under IG.

(2) To allow IG users to make calls to the higher-level subroutines in PDS.

To use the IG-PDS interface, a program must be executed as follows:

```
$RUN objectfile+*IG+*PLOTSYS
```

The files *IG and *PLOTSYS must be concatenated in this order to the object file of the program. The file *IG contains several subroutines that have the same names as the following PDS subroutines:

```
PLTBGN
PLTEND
PENDN
PENUP
PLINE
etc.
```

Any calls to these subroutines are intercepted by the corresponding IG subroutines, which operate on the IG data structure instead of generating a plot description. This data structure may then be displayed on the terminal screen and modified interactively.

RUNNING PDS PROGRAMS INTERACTIVELY

A "PDS program" is one that primarily calls PDS subroutines. The first such call generates an implicit call to PLTBGN (this call need not be explicit). Following this, there will be a sequence of calls to other PDS subroutines, terminated by a call to PLTEND.

The default IG-PDS interface is designed so that PDS programs may be run under IG with no modification, simply by concatenating *IG+*PLOTSYS to the object file. No explicit calls to IG subroutines need be made. Each call to PLTEND causes the plot to be displayed on the terminal screen. Then, the user is prompted with the following question:

Blow-up, Redraw, Plot, or Continue?

A response of B (for Blow-up) allows the user to expand a rectangular subregion of the plot to view it in more detail. The user is prompted first for the position of the lower-left corner and then for the position of the upper-right corner. These positions are entered via the graphic input facility of the terminal (see the section "Graphic Input"). This rectangular subregion is then expanded to fill the entire screen.

A response of R (for Redraw) causes the plot to be redisplayed at its original scale. In other words, the effect of a previous blow-up is reversed.

A response of P (for Plot) causes a plot description to be written on logical I/O unit 9, just as if the program had been run without *IG.

A response of C (for Continue) causes a return from PLTEND to the calling program.

After the program has been tested interactively, it may be rerun with only *PLOTSYS concatenated to the object file. In this case, each call to PLTEND will generate a normal plot description.

THE DEFAULT IG-PDS INTERFACE

To make effective use of the IG-PDS interface, the programmer should understand how calls to the PDS subroutines affect the IG data structure. The following paragraphs describe the default actions of these subroutines.

(Implicit) calls to PLTBGN are intercepted by the IG version of PLTBGN. This subroutine implicitly calls IGINIT (if necessary) to initialize the IG data structure. It then activates a subpicture named 'PLOT' by making the following call:

```
CALL IGBGNS('PLOT')
```

Calls to lower-level, pen-moving PDS subroutines are intercepted by the corresponding IG subroutines. These add lines and text to the active subpicture, which by default is 'PLOT'. Coordinates are given in terms of the plotter coordinate system (inches of plotter paper).

Calls to higher-level PDS subroutines are handled by the subroutines themselves. However, these in turn call the lower-level PDS subroutines. In this way, the higher-level PDS subroutines also add lines and text to the active subpicture.

Calls to PLTEND are intercepted by the IG version of PLTEND. This subroutine in turn calls IGENDS('PLOT'). It then determines the minimum and maximum X and Y values in 'PLOT' and performs a window-to-viewport mapping such that 'PLOT' will fit entirely on the screen. It then calls IGDRON('TERMINAL') to display 'PLOT' and enters the "Blow-up, Redraw, Plot, or Continue?" loop. This loop may be disabled by making the following call:

```
CALL IGPDSW(1,0)
```

Subsequent calls to PLTEND will result in a calls to IGENDS('PLOT') and IGDRON('TERMINAL') followed by an immediate return to the calling program.

GENERATING A REAL CALCOMP PLOT

When using the IG-PDS interface, calls to PDS subroutines result in additions to the IG data structure. This data structure is then displayed on the terminal screen. The user program may generate a real CalComp plot corresponding to the screen image by making the following call:

```
CALL IGDRON('CALCOMP')
```

A plot description is written on logical I/O unit 9 or on a file or device specified by the user (see Appendix K). The plot is scaled to fit an 8.5x11-inch sheet of plotter paper. Note that the coordinates of the final plot are several steps removed from the coordinates that were initially passed to the lower-level, pen-moving PDS subroutines. However, the size and placement of the plot may be altered by calling the IGCTRL subroutine (see Appendix K).

CALLING PDS SUBROUTINES FROM IG PROGRAMS

An "IG program" is one that primarily calls the IG subroutines. Such a program may make incidental calls to PDS subroutines for the purpose of generating coordinate axes, grids, curves, etc. To do this effectively, the program should be structured as follows:

(1) The program should call IGINIT before calling any other IG (or PDS) subroutine.

(2) Before making any calls to PDS subroutines, the program should make the following call:

```
CALL IGPDSW(0,0)
```

This disables the default IG-PDS interface. Calls to PLTBGN (implicit) and PLTEND are no longer translated into calls to IG subroutines. In fact, calls to PLTBGN and PLTEND should be omitted.

(3) The program should create and activate one or more separate subpictures (via IGBGNS, etc.) to hold lines and text to be generated by PDS subroutines.

(4) Calls to lower-level, pen-moving PDS subroutines are still intercepted by the corresponding IG subroutines. These add lines and text to the active subpicture. The higher-level PDS subroutines implicitly call the lower-level PDS subroutines.

(5) The PDS subroutines generate lines and text in the plotter coordinate system. This may be mapped into the -1 to +1 screen coordinate system by making the following call:

```
CALL IGTRAN(NAMSUB,'MOVE',-XMX/2,-XMX/2,'SCALE',2/XMX)
```

Here, NAMSUB is the subpicture involved and XMX is either the maximum X extent as specified by a call to PLTXMX or the default of 36.

(6) The program is responsible for calling IGENDS for the subpictures involved and for calling IGTRAN to move and scale them so they will fit on the screen.

(7) The program is responsible for calling IGDRON('TERMINAL') to display the data structure on the screen.

December 1980

APPENDIX K: GENERATING CALCOMP PLOTS

A plot description is a set of instructions that can be used to drive the CalComp plotter. A PDS file is a file (on disk or tape) that contains one or more plot descriptions.

The mere existence of a PDS file is not enough to generate a CalComp plot. Rather, a plot request must be entered into a special system queue. This plot request will specify a PDS file to be read, a scale factor to be used, and so on. Plot requests may be entered into or deleted from the queue by means of the public file program *CCQUEUE. When a request is entered, a six-digit receipt number is issued. This number may later be used to claim the plot at the output window.

GENERATING PLOT DESCRIPTIONS

One way to generate plot descriptions is to use the Plot Description System (PDS). This is a package of subroutines which may be used to generate coordinate axes, grids, bar graphs, line graphs, and so on. A single plot description is generated by a series of subroutine calls ending with a PLTEND call. See Appendix J and MTS Volume 11, Plot Description System.

The IG system may also be used to generate plot descriptions. These will be written on logical I/O unit 9, which may be assigned in the $RUN command for the user program:

```
$RUN objectfile+*IG 9=pdsfile
```

This specifies that plot descriptions are to be written in the file "pdsfile". Alternatively, the user program may make the call

```
CALL IGCTRL('CALCOMP','PFILE','pdsfile ')
```

to specify that plot descriptions are to be written in "pdsfile". Note that the file name must be followed by a trailing blank. Typically, the user program will call IGDRON('TERMINAL') to generate a screen image that represents the current data structure. It will then ask the user whether he or she wishes to generate a plot description, and if so, it will make the following call:

```
CALL IGDRON('CALCOMP')
```

This will generate a single plot description. The resulting plot will be similar to the screen image. However, there may be some differences in the interpretation of pen numbers and the printing of text.

Whenever IGDRON generates a plot description, it prints the message PDS: PLOT DESCRIPTION GENERATION BEGINS. This message may be turned off (but not turned back on) by making a call of the form:

```
CALL IGCTRL('CALCOMP','PHDR',0)
```

By default, the plot will be scaled to fit an 8.5x11-inch sheet of plotter paper. The -1 to +1 visible region (corresponding to the screen) will be mapped into a 7.5x7.5-inch square, with an X margin of 0.5 inches and a Y margin of 1.75 inches. A different scale may be specified for future plots by making the following call:

```
CALL IGCTRL('CALCOMP','SIZE',S)
```

In future calls to IGDRON('CALCOMP'), the -1 to +1 visible region will be mapped into a square S inches on a side with the same margins as above: an X margin of 0.5 inches and a Y margin of 1.75 inches. Different margins may be specified for future plots by making the following calls:

```
CALL IGCTRL('CALCOMP','XMAR',XMARGN)
CALL IGCTRL('CALCOMP','YMAR',YMARGN)
```

The maximum size of the plotter paper is 360 inches in the X dimension and 33 inches in the Y dimension. If a plot is scaled larger than this, a portion of the visible region will be clipped off. The first parts to be sacrificed will be the blank margins, followed by the edges of the plot. The center of the plot will always be visible. For example, if the S value above is set to 40.0, the entire X dimension will be visible but the top and bottom 3.5 inches of the Y dimension will be clipped off.

QUEUEING PLOT REQUESTS

Plot requests may be queued by means of the public file program *CCQUEUE. For convenience, the description of this program is reprinted on the following pages.

*CCQUEUE

Contents: The public file *CCQUEUE contains the queueing program for requesting postprocessing of user plot descriptions.

Purpose: To place the user's plotting request(s) on the queue for plotting.

Use: This program is invoked by the $RUN command.

Program Key: *CCQUEUE

Logical I/O Units Referenced:

SCARDS - queueing requests.
SPRINT - plot receipt numbers and prompting messages for queueing requests.
SERCOM - error comments.
GUSER - user responses to error comments.

Parameters: A single queueing request may be specified in the PAR field of the $RUN command, if no requests are to be read from SCARDS.

Description: The complete description of the plotting system is given in MTS Volume 11, Plot Description System.

Each queueing request specifies a file or tape containing one or more plot descriptions. *CCQUEUE reads the plot descriptions, determines the cost of plotting, places the request in the system plotting queue, and charges the user for the plotting costs. At each scheduled plotting time, the system postprocessor removes the request from the queue, reads the user's file (or tape), and produces commands for the plotter.

*CCQUEUE prompts for queueing requests using the message "ENTER PLOT REQUEST". The response may be a queueing request for a file or tape, a line consisting of the string "MTS", a line beginning with a "$", or an end-of-file indicator. If the queueing request is for a file, the request should consist of the file name, optionally followed by one or more blanks and a scale factor. Explicit concatenations are permitted, as are line number ranges. Only permanent files may be queued--temporary files may not be queued.

The response "MTS" causes a return to MTS command mode. A response that begins with a dollar sign is treated as an MTS command that is to be executed immediately. For either response, "MTS" or an MTS command, execution of

*CCQUEUE may be resumed by entering a $RESTART command (unless an MTS command causes *CCQUEUE to be unloaded). An end-of-file response terminates execution of *CCQUEUE.

If the request is for a magnetic tape, the tape should already be mounted. The request should contain first the pseudodevice name, followed by the tape ID. Parameters must be separated by one or more blanks and/or commas. There are three optional parameters. One is the scale factor, which may appear anywhere after the pseudodevice name. Another is the FILES parameter, which may also appear anywhere after the pseudodevice name. This takes the form FILES=n, where "n" is the number of files to be plotted. The default is FILES=1. Finally, the POSN parameter may appear anywhere after the tape ID. This takes the form, POSN=string, where "string" specifies a file on the tape (any string legal for the POSN control command). If this parameter is not specified, *CCQUEUE begins reading at the current tape position (even if that is in the middle of a file). (At postprocessing, the tape will be mounted and positioned appropriately, even to the middle of a file.) *CCQUEUE and the postprocessor use whatever blocking is in effect at the time the queueing request is entered. *CCQUEUE does not rewind the tape after reading it. Pool tapes may not be queued.

Any plot request may include a scale factor. The scale factor will be applied to each point of the plot. This parameter takes the form SCALE=x, where "x" is a positive integer, F-type, or E-type real number.

Users can request delivery of plots to another station by specifying the DELIVERY=station parameter on the queue request, e.g., DELIVERY=NUBS. The DELIVERY parameter is effective for only one queue request. If the DELIVERY parameter is not specified, the setting of the MTS $SET DELIVERY option will be used (which defaults to the Computing Center).

Users can request "quick" plotting by specifying the ASAP parameter on the queue request. This will provide a faster than normal service. Ballpoint pens will be used instead of liquid ink and the appearance of the plot will differ; lines will be thinner and not as dark, and dots may not be drawn very well. The plotter will use only red, blue, and black ballpoint pens, and pen size changes will not be allowed. See the public file *PLOTTIMES in MTS Volume 2, Public File Descriptions, for the schedule of ASAP plotting service.

Error Processing:

If *CCQUEUE encounters an error in the user's plot description file, it prints a comment describing the error, together with the file name and line number where the error was discovered. If an error is encountered in batch mode, reading of the current plot description stops, the plot is not queued, and *CCQUEUE begins processing the next request, if any. In conversational mode, the user is asked if he wishes *CCQUEUE to continue. The legal responses are YES (or OK) to continue processing (taking the error recovery action described below), and NO, to stop processing at this point (this does not abort the plot). The user may also enter MTS or an MTS command preceded by a "$", in which case control will be returned to MTS. (The user may restart using $RES, and he will again be prompted for a response.) The error recovery actions are

Page Revised March 1983 December 1980

Error	Action Taken by *CCQUEUE
Record not a plot record	Record ignored
Actual record length not equal to expected length	Record ignored
Missing PBGN	PBGN (logically) inserted with default normalizing factor
Invalid PSYM record	Record ignored
Invalid PPEN record	Record ignored
Missing PEND	PEND (logically) inserted

If the user indicates that processing should continue after an error is encountered, and then queues the plot, the same error recovery action will be taken by the postprocessor.

A line beginning $CONTINUE WITH ... will be treated as an invalid plot line. Implicit concatenation may not be used in plot description files (or tapes).

Queueing:

After *CCQUEUE finishes reading the plot description (because it has reached the end or the user has told it not to continue after an error), it prints the number of plots (blocks), the plotting time required, the amount of paper required, and the cost. In conversational mode, the user is asked whether the plot should actually be queued. If he replies NO, the plot is aborted and he is prompted for a new plot request (unless the plot request was taken from the PAR field). If he replies YES, or if *CCQUEUE is being executed in batch mode, the plot request is placed in the queue, he is charged for the plot, and a receipt number is printed (this receipt number must be used to obtain the plot at the output window later). In addition, if the plot description was contained in one or more disk files, *CCQUEUE will, if necessary, permit the queued files so that the postprocessor, which is a program with PKEY=*CCQUEUE, may access the plot descriptions. *CCQUEUE will establish READ access for the file to the postprocessor if necessary. (Specifically, the file will be permitted "READ PKEY=*CCQUEUE" unless the postprocessor already has read access. See the description of the $PERMIT command in MTS Volume 1, Michigan Terminal System.) A message is printed for each file so permitted. If *CCQUEUE cannot determine the permit status of a file, a message is printed, and the user must be sure that the file is

permitted before the plot request is postprocessed. See the section below on postprocessing.

The plot will be aborted if it is null, if the user's plotting time limit (local, global, or ID) has been exceeded, if the user indicated abort after an attention interrupt (see below), or if reading stopped on an error while in batch mode.

Cancellation:

A queued plot may be canceled by entering CANCEL nnnnnn where "nnnnnn" is the receipt number, whenever *CCQUEUE is prompting for a plot request. A plot can be cancelled only by the signon ID that queued it. If the plot specified is still in the queue (i.e., it has not been postprocessed), it will be cancelled and a message to that effect will be printed. Plotting charges will be rebated automatically at a later time.

Attention Interrupt Processing:

If the user issues an attention interrupt, processing of the current plot request is suspended; the subsequent action taken by *CCQUEUE is as follows: (1) If it was parsing a plot request, the plot request is discarded and the user is prompted to enter a new request; (2) *CCQUEUE will print "READING:" or "QUEUEING:", depending upon whether it was in the process of reading the plot description or queueing the plot. In either case, if the user enters a null line or an end-of-file, *CCQUEUE will resume processing the current request. If the user enters "MTS" or an MTS command beginning with a "$", *CCQUEUE will return to MTS command mode; if a command was given, it will be immediately executed by MTS. If the user enters anything else, the plot request will be aborted.

If the user issues a second attention interrupt while *CCQUEUE is processing the first interrupt, an immediate return is made to MTS command mode. The user may subsequently reenter *CCQUEUE via the $RESTART command.

Postprocessing:

After a plot request has been queued, the user's plot description must be available for reading by the system postprocessor at the scheduled plotting time. If the plot description is in a file, the file must be permitted, and it must not be locked at any level higher than READ. If the file is not permitted, the plot will be canceled; if it is locked, the plot request will remain queued until the next scheduled plot time at which

it is accessible. If the plot description is on a tape, the tape must be available for mounting (e.g., not already mounted); otherwise, the plot will remain queued until the tape is available at a scheduled postprocessing time. If the tape ID is incorrect, the plot will be canceled.

Examples: In the following examples, terminal output appears in uppercase and user input appears in lowercase.

In the first example, the plots contained in two files, PLOTS and MOREPLOTS, are to be queued. At line 2 in MOREPLOTS, *CCQUEUE discovers that a PBGN record seems to be missing, and asks the user if it may assume a PBGN record with a default normalizing factor. The user OKs this, so *CCQUEUE continues reading the plot file. At line 2.5, *CCQUEUE discovers a PPEN record with an invalid color, and asks the user whether it may ignore the line. The user invokes the file editor to examine the line, and then tells *CCQUEUE not to continue. The user then OKs queueing of all plots up to line 2.5 of MOREPLOTS.

```
#$r *ccqueue
 EXECUTION BEGINS
 .
 .
 ENTER PLOT REQUEST:
 plotfile
   2 PLOTS; PLOTTING REQUIRES  94 SEC. AND 19 IN.; $.31
   PEN WAS UP 34% OF THE TIME
 OK?
 ok
 PLOT ASSIGNED RECEIPT # 516061
 ENTER PLOT REQUEST:
 plots+moreplots
 MISSING PBGN RECORD.
 DISCOVERED AT LINE      2.000 IN MOREPLOTS
 OK TO USE DEFAULT?
 y
 INVALID PEN TYPE.
 DISCOVERED AT LINE      2.500 IN MOREPLOTS
 OK TO IGNORE?
 $ed moreplots
#$ED MOREPLOTS
:p 2.5
:     2.5    PPEN??PINK
:stop
 OK TO IGNORE?
 n
 PLOT UP TO LAST ERROR CAN BE QUEUED.
   2 PLOTS; PLOTTING REQUIRES  47 SEC. AND 9 IN.; $.17
   PEN WAS UP 41% OF THE TIME
```

```
OK?
ok
PLOT ASSIGNED RECEIPT # 516064.
.
.
```

In the second example, the user does not have appropriate access to XXXX:PLOTS for *CCQUEUE to determine its permit status. Note that the user must make sure that the file is permitted before the next postprocessing time.

```
#$r *ccqueue
 EXECUTION BEGINS
 .
 .
 ENTER PLOT REQUEST:
 xxxx:plots
   1 PLOTS; PLOTTING REQUIRES 35 SEC. AND 7 IN.; $.13
   PEN WAS UP 61% OF THE TIME
 OK?
 y
 **PERMIT STATUS OF "XXXX:PLOTS      " UNKNOWN.
 PLOT ASSIGNED RECEIPT # 516068.
 .
 .
```

In the last example, the file whose data set name (DSN) is PLOTS and the file following it on tape *I* are to be queued. The user allows *CCQUEUE to position the tape. These two files together contain three plots. *CCQUEUE encounters two errors, and the user tells it to continue after each.

```
#$r *ccqueue
 EXECUTION BEGINS
 .
 .
 ENTER PLOT REQUEST:
 *I* 'CALCOMP PLOTS' p=PLOTS files=2
 MISSING PBGN RECORD.
 DISCOVERED AT LINE      14.000 IN *I*
 OK TO USE DEFAULT?
 y
 INVALID PEN TYPE.
 DISCOVERED AT LINE      78.000 IN *I*
 OK TO IGNORE?
 y
   3 PLOTS; PLOTTING REQUIRES 117 SEC. AND 27 IN.; $.47
   PEN WAS UP 28% OF THE TIME
 OK?
 y
 PLOT ASSIGNED RECEIPT # 516070.
 .
```

APPENDIX L: OBSOLETE SUBROUTINES

The subroutines described in this appendix are considered obsolete. They will remain in IG indefinitely, but no maintenance will be performed for them. In general, the functions of these subroutines have been taken over by other subroutines, as described in the following list:

Obsolete Subroutine	Replacement
IGATTR(namsub,attrib,atval)	IGATTS(namsub,attrib,atval)
IGHUE(namsub,color)	IGATTS(namsub,'PEN ',penno)
IGINT(namsub,intens)	IGATTS(namsub,'PEN ',penno)
IGTEXT('textstring',length)	IGFMT('textstring','A',length)

The IGHUE and IGINT subroutines are described in more detail on the following pages. These descriptions are provided to aid in deciphering or modifying existing programs. These subroutines should not be used to write new programs.

IGHUE

Purpose: To specify the color of a subpicture.

Calling Sequence:

FORTRAN: CALL IGHUE(namsub,color)

Parameters:

namsub is the name (four-character or internal) of the subpicture whose color is being specified.

color is a character string of at least four characters specifying the color of namsub.

Description: This subroutine specifies the color which IGDRON will assign to the subpicture. IGHUE may be called at any time, whether the subpicture is active or not. The colors supported by a particular device are listed in Appendix I. If a color is not supported by a device, subpictures having that color will be displayed in the default color for the device. For example, the CalComp 936 plotter supports three colors, 'BLACK', 'RED ', and 'BLUE'. 'BLACK' is the default color. If an 'ORANGE' subpicture is passed to the CalComp 936 plotter device-dependent routine, it will be plotted in 'BLACK'. If color is specified as 'DEFAULT', namsub will inherit its color from the next-higher-level picture.

IGINT

Purpose: To specify the intensity of a subpicture.

Calling Sequence:

FORTRAN: CALL IGINT(namsub,intens)

Parameters:

namsub is the name (four-character or internal) of the subpicture whose intensity is being specified.

intens is either an integer value between 0 and 255 that specifies the new intensity, or the character string 'DEFAULT'.

Description: This subroutine specifies the intensity which IGDRON will assign to the subpicture. IGINT may be called at any time, whether the subpicture is active or not. When IGDRON is called, the subpicture will be displayed at an intensity as close as possible to intens. However, if intens is the string 'DEFAULT', the subpicture will inherit its intensity from the next-higher-level picture.

December 1980

INDEX

Reader’s Comment Form

Integrated Graphics System
Volume 17
December 1980
(September 1984 Reprint)

Errors noted in publication:

Suggestions for improvement:

Your comments will be much appreciated. The completed form may be sent to the Computing Center by Campus Mail or U.S. Mail, or dropped in the Suggestion Box at the Computing Center, NUBS, or UNYN.

Date _____

Name _____

Address _____

_____

_____

Publications
Computing Center
University of Michigan
Ann Arbor, Michigan 48109

Update Request Form

Integrated Graphics System
Volume 17
December 1980
(September 1984 Reprint)

Updates to this manual will be issued periodically as errors are noted or as changes are made to MTS. If you desire to have these updates mailed to you, please submit this form.

Updates are also available in the memo files at the Computing Center, NUBS, and UNYN; there you may obtain any updates to this volume that may have been issued before the Computing Center receives your form. Please indicate below if you desire to have the Computing Center mail to you any previously issued updates.

Name _____

Address _____

_____

_____

Previous updates needed (if applicable):_____

The completed form may be sent to the Computing Center by Campus Mail or U.S. Mail, or dropped in the Suggestion Box at the Computing Center, NUBS, and UNYN. Campus Mail addresses should be given for local users.

Publications
Computing Center
The University of Michigan
Ann Arbor, Michigan 48109

Users associated with other MTS installations (except the University of British Columbia) should return this form to their respective installations. Addresses are given on the reverse side.

Addresses of other MTS installations:

Publications Clerk
352 General Services Bldg.
Computing Services
The University of Alberta
Edmonton, Alberta
Canada T6G 2H1

Information Officer, NUMAC
Computing Laboratory
The University of Newcastle upon Tyne
Newcastle upon Tyne
England NE1 7RU

Rensselaer Polytechnic Institute
Documentation Librarian
310 Voorhees Computing Center
Troy, New York 12181

Simon Fraser University
Computing Centre
User Services Information Group
Burnaby, British Columbia
Canada V5A 1S6

Wayne State University
Computing Services Center
Academic Services Documentation Librarian
5950 Cass Ave.
Detroit, Michigan 48202