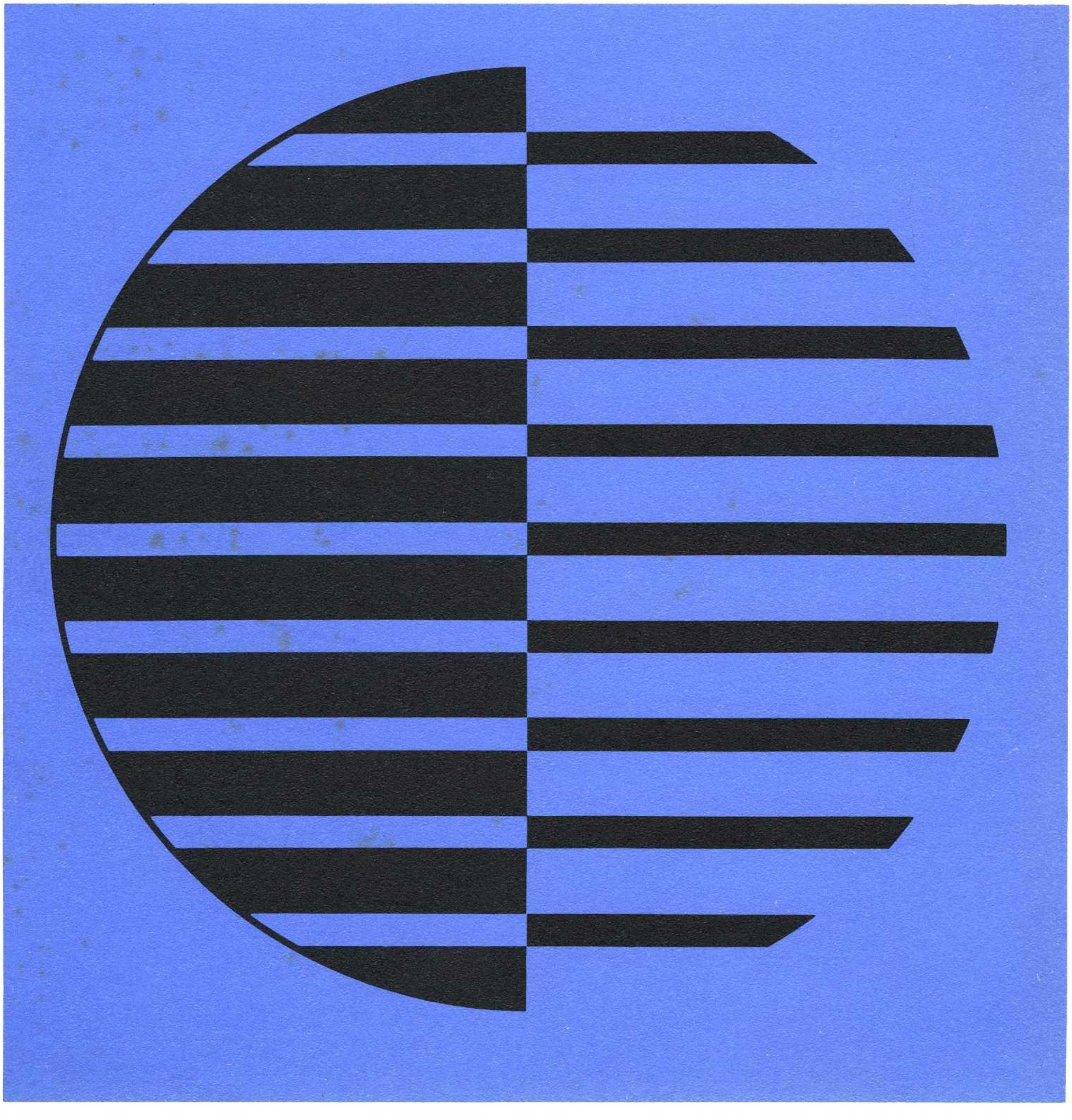
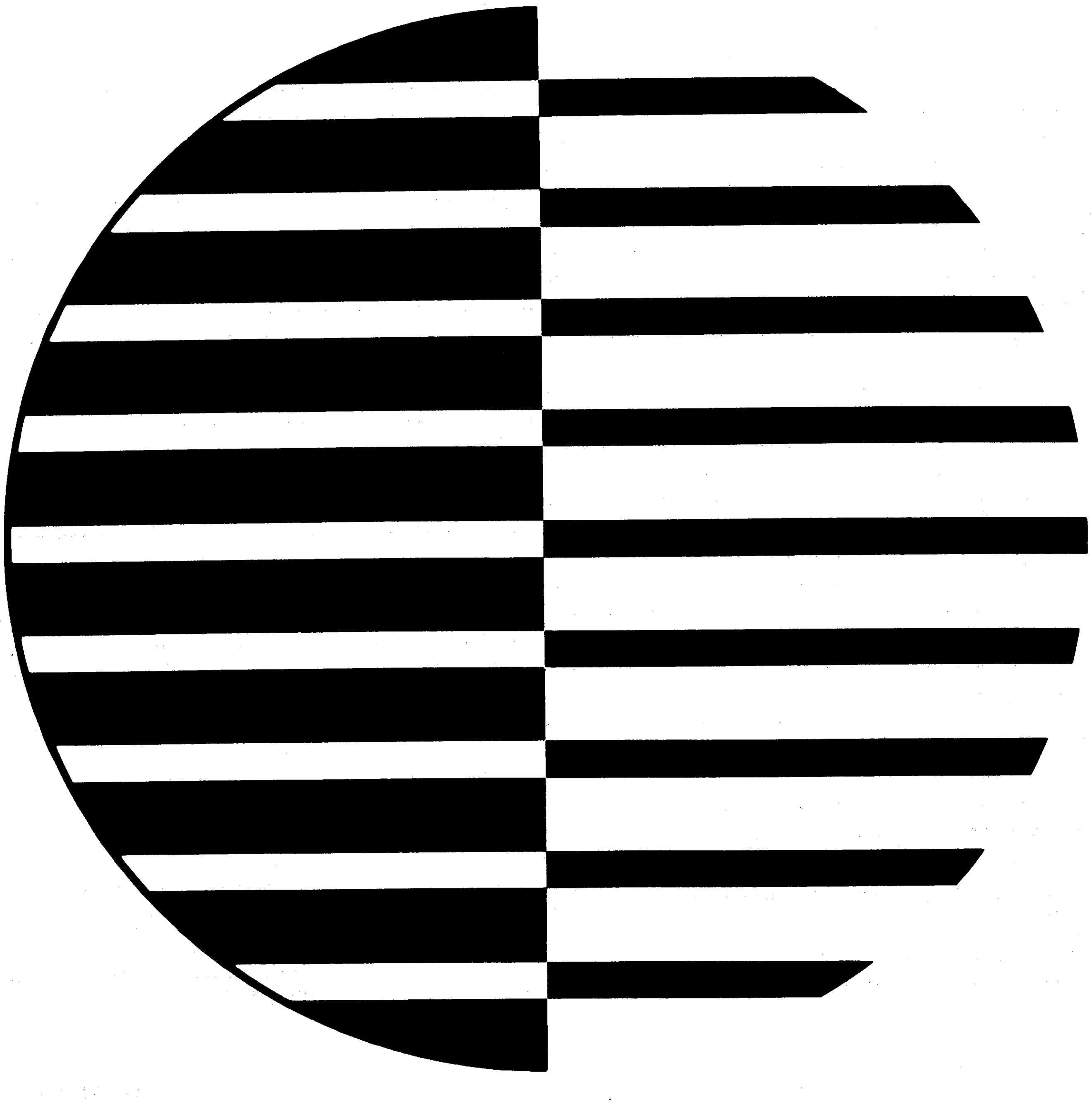
CONTROL DATA®
6000 SERIES COMPUTER SYSTEMS
274 INTERACTIVE GRAPHICS SYSTEM
Users Guide



CONTROL DATA®
6000 SERIES COMPUTER SYSTEMS
274 INTERACTIVE GRAPHICS SYSTEM
Users Guide



REVISION RECORD

REVISION	DESCRIPTION
A	Original release.
(10/1/70)	
B	Various minor editorial corrections made by authority of PCO CF 1067, and Appendix G added.
(1/1/71)	
C	Certain additions incorporated to make the manual compatible with SCOPE 3.3, made by authority
(1/1/71)	of PCO CF 1069.
D	Manual completely retyped and reissued to reflect the final form of the Version 2 software under
(4/1/71)	SCOPE 3.3. Revision lines have been omitted because of the magnitude of the changes.
Publication No. 44629300	

Additional copies of this manual may be obtained from the nearest Control Data Corporation sales office.

6000 Series 274 Interactive Graphics System
User's Guide

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Systems Publications
215 Moffett Park Drive
Sunnyvale, California 94086

or use Comment Sheet in the back of this manual.

PREFACE

This manual explains the use of the CONTROL DATA® 6000 series 274 Interactive Graphics System, Version 1 and includes conversion information for 274 IGS Version 2. It provides graded examples which illustrate the capabilities of the system. Review questions and sample coding problems are presented as a tutorial aid. An extensive knowledge of programming is not required, but familiarity with the hardware and software used in the system would be useful.

For more information related to the system's software, see the following Control Data publications:

Title	Publication Number
6000 Series SCOPE 3.3 Reference Manual	60305200
6000 Series SCOPE 3.3 Operating Guide	60306400
6000 Series Systems Reference Manual	60100000
6000 Series Interactive Graphics System General Information Manual	44616700
6000 Series EXPORT/IMPORT High-Speed and 274 Interactive Graphics System Operator's Guide	17303100
6000 Series 274 Interactive Graphics System Reference Manual	17303600
1700 Series 1744 Digigraphics Controller Reference/Customer Engineering Manual	60283300
General Description of the 274 Display Console and the 1744 Controller	17223000
6000 Series SCOPE 3.2 Reference Manual	60189400
6000 Series SCOPE 3.2 Operating Guide	60179600
6000 Series 274 Interactive Graphics System Version 1 Reference Manual	44616800
6000 Series 274 Interactive Graphics System Version 1 BATCHIO and EXPORT/IMPORT Operator's Guide/Installation Handbook	44628900

CONTENTS

APPENDICES

FIGURES

TABLES

INTRODUCTION 1

The CONTROL DATA® 6000 series 1700/274 Interactive Graphics System allows a graphics console operator to access a large computer for real-time use in a dual graphics/batch processing mode while maintaining full machine capability.

To accomplish this, a separation of functions must be established. Graphics programming is done only on the 6000 series computer, while the 1700 series computer software operates without programmer intervention. Graphics programs are written in standard FORTRAN. They are externally independent of display hardware characteristics. Graphics programs run at a dedicated Graphics control point.

The Interactive Graphics software operates as two separate but communicating groups of routines — one in the 6000 series computer (which time-shares central memory), the other in the 1700 series computer.

This user's guide presents five problems of increasing complexity which demonstrate the capabilities of the graphics console and provide graded lessons on the use of the system. Each IGS call is presented in the context of one of these sample problems. Various appendices present additional examples of coding, reference and review tables, coding hints, and an answer key.

6000 SERIES COMPUTER SOFTWARE

The 6000 series computer software consists of the SCOPE operating system (expanded to include graphics features), a FORTRAN compiler, the Basic Graphics Package, the scheduler (for time-sharing the graphics programs), and a package for communication between graphics programs and the 1700 series computer.

1700 SERIES COMPUTER SOFTWARE

The 1700 series computer software consists of an IMPORT program to handle 6000-1700 series data communications, a Mass Storage Operating System (MSOS) to drive printers, readers and punches, a buffer translator which translates 6000 series data into display-oriented data and vice versa, and a graphics package which processes interrupts and data from the consoles and acts as a driver for the graphics consoles.

GRAPHICS HARDWARE

The graphics system provides an interface for the handling of graphic or alphanumeric information. Entries or modifications are made at the console. These entries are placed into the 1700 series computer in digital form and become available for use by the 6000 series system. This graphics input is visible on the cathode ray tube and can be used for information processing by an applications program under console operator control. The results of such processing are immediately displayed on the screen. The static display of graphic and alphanumeric data is provided by buffers so that the consoles are essentially off-line devices.

GRAPHICS CONSOLE

The user's input/output and control center is the graphics console. The major portion of system graphics capability can be controlled from the console.

The console cabinet is a desk-size unit which mounts a rectangular housing assembly, off-centered to the left, and provides a writing surface to the right. The housing assembly contains a magnetic shield and a 21-inch diameter cathode ray tube centered on the front panel housing. The display items on the cathode ray tube screen can be easily seen at normal light levels.

The cathode ray tube has a nearly flat display surface to minimize parallax error. The tube is equipped with an implosion shield for the protection of the operator, and is coated with a two-layer P-7 phosphor. One layer produces blue-violet light with a short persistence to facilitate light-pen tracking. The other layer produces yellow-green light and has a longer persistence to eliminate flicker. With a continuously refreshed display, the light from both phosphor components combines to appear light blue to the human eye. The cathode ray tube has 314 square inches of surface available for display.

Data can be entered on the cathode ray tube via the light-pen and the two optional keyboards.

CONTROLS

The controls available to the console operator include the keyboards, light-pen, light registers, and light buttons. The light registers and light buttons are defined by the application program and formatted for display on the screen by the 1700 Basic Graphics Package routines.

FUNCTION KEYBOARD

The 16-key function keyboard informs the application program that an operation is requested (Figure 1-1).

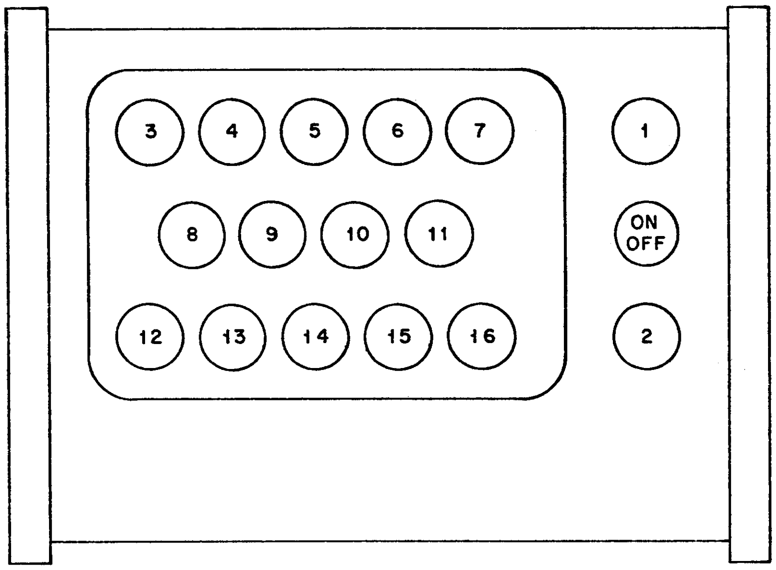


Figure 1-1. Function Keyboard

Fourteen keys contain snap-action switches that remain on after an initial press and off after being pressed again; the other keys must be held down to give an "on" status. Each key has an internal light that shows the operator when it is on. Removable plastic cards may be placed over the keys to label the function of each. All keys can be given new functional assignments by the application program through the 6000 Basic Graphics Package.

ALPHANUMERIC KEYBOARD

The alphanumeric keyboard (Figure 1-2) provides typewriter-like symbolic input to the application program. The keyboard layout is similar to that of a conventional teletypewriter. Each key enters an 8-bit ASCII character code in the left-hand portion of a status word that is fetched by the 1700 Basic Graphics Package. The characters are collected into line images and displayed on the 274 Console screen in the currently defined light register.

LIGHT-PEN

The light-pen has two functions: tracking and picking. Tracking may be used to place a light source (the tracking cross) at any desired position on the console screen so that a graphics display item[†] may be created there, or to designate that position as an area of

[†]A graphics display item is any item displayed on the 274 Console.

274 GRAPHICS CONSOLE 2

INTRODUCTION

The first part of this section describes the 274 Graphics Console in terms of addressing the screen (grid) coordinates, the working and control surfaces available, and character sizes.

The rest of the section is devoted to an explanation of the display capability of the graphics system/console. A graphics program which displays a winding road traffic sign[†] is explained and the three essential steps used to create this display are discussed in detail.

DISPLAY GRID

This system uses a 274 Graphics Console. Its display screen (Figure 2-1) consists of a circle divided into addressable grid coordinates on the x-axis (horizontal) and the y-axis (vertical) called digigraphic units (dgus). There are approximately 200_{10} dgus per inch on the 21-inch diameter screen. The screen has absolute center coordinates (0, 0) with the x-axis endpoint coordinates at (-2048, 0) and (+2047, 0) and the y-axis endpoint coordinates at (0, 2047) and (0, -2048). All points on the screen are addressable in octal notation (400B, 1400B) or decimal notation (256, 768), but the x-coordinate and the y coordinate must never be greater than 2047 or less than -2048. (All subsequent references to coordinate points will be in decimal notation.)

The grid is actually larger than the screen so that all points on the screen can be addressed. Points beyond the edge of the screen can be addressed by a programmer, but are invisible to a user directly in front of the screen. If viewed at an angle, such points can be seen reflected off the side of the tube.

SCREEN ORGANIZATION

When planning a graphics program, the programmer should consider the following:

- How large will the working surface be
- Will items be displayed as ignore, single pick, string pick, or button pick items
- Does the program require a font for input
- Should the tracking cross be turned on

[†]Appendix A includes a different approach to the Winding Road problem.

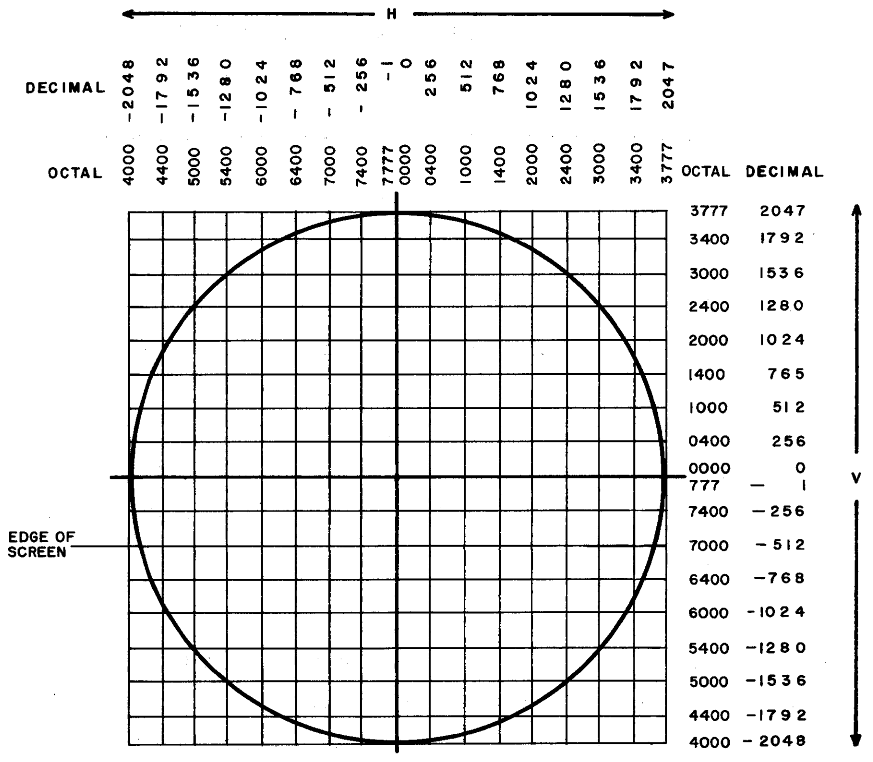


Figure 2-1. Display Grid System

The organization of the screen is completely up to the programmer. However, certain conventions may be used for a wide variety of applications. These conventions allow a programmer to make maximum use of the screen area, yet help him avoid addressing grid coordinates off the screen.

WORKING SURFACE

One convention is to divide the screen into a working surface and a control surface. The working surface is reserved for the display of graphics forms and is contained within a frame defined by the programmer. The frame may or may not be displayed.

CONTROL SURFACE

The control surface is defined as the area outside the frame or frames and is normally reserved for light buttons, light registers, and the tracking cross (when it is not in use on the working surface). Figure 2-2 shows a sample of one type of screen organization.

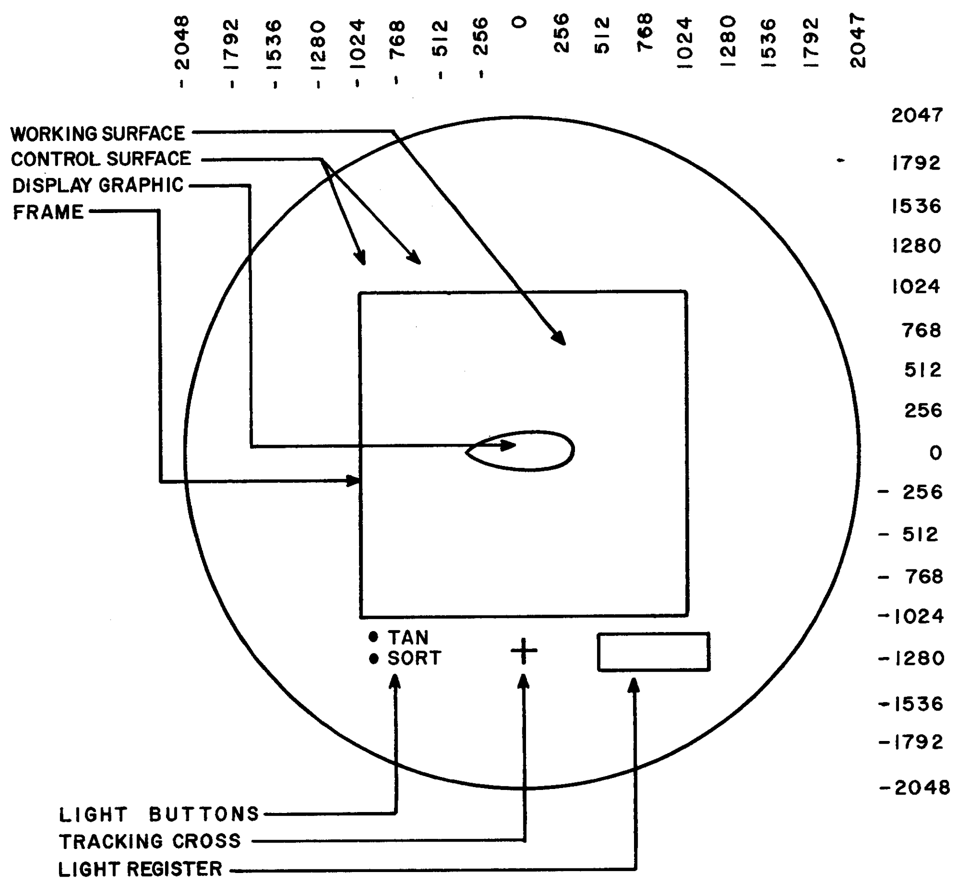


Figure 2-2. Sample Display Surface Organization

CHARACTER SIZE

When alphanumeric information is to be displayed on the screen, an idea of the character size can be obtained by considering each character to be a 24 by 24 dgu square (with spacing included at the top and right side of this square). For instance, if the user specifies that an A will be displayed at coordinates (256, 768), the lower left-hand corner of the 24 by 24 dgu square is positioned at (256, 768).

WINDING ROAD

A prime feature of the graphics system is its picture and information displaying ability. To illustrate how a picture is created, a simple depiction of the lines, descriptive wording, and distinctive shape that form the winding road traffic caution warning sign is shown in Figure 2-3. The working surface is the entire screen. Items will be displayed as ignore items (which means they are insensitive to the light pen). Neither font nor tracking cross are used.

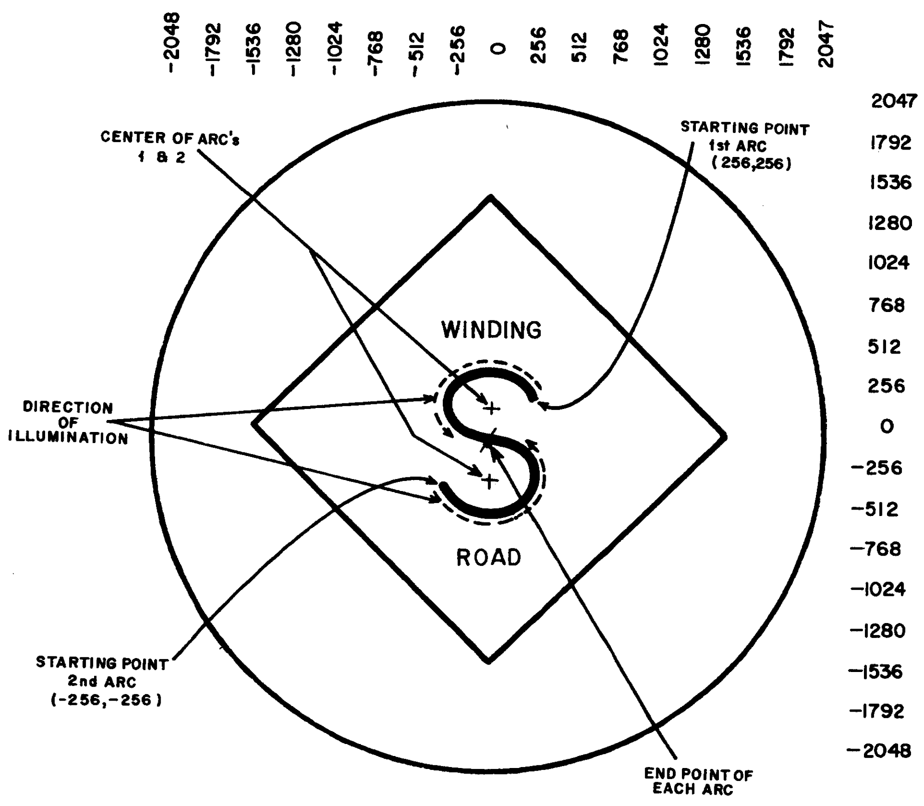


Figure 2-3. Winding Road

Standard IGS lines have an approximate width of 5 dgus at 50% beam intensity. Assume that arrays are dimensioned and that masks are set.[†]

The pick will be ignored if the user picks an item with IDDT = 1 when the light-pen is attached to the console. If the user picks several display items on the console with a light-pen, only the last one designated as a single pick item with IDDT = 2 will remain on an internal queue. This is in contrast to string pick items where each item in a string of light-pen picks is queued. Later, IGS calls will retrieve the queued items.

In the Winding Road example, the number of the console referenced (NCON) is 1.

FIRST STEP IN CREATION

POSITIONING THE BEAM

Position the light beam on a point which will become a corner of the diamond-shaped perimeter of the sign. (The beam could be positioned at any point on the perimeter.) For this

[†]If all items are ignore items with an IDDT = 1, the programmer need not use GIMASK to set a mask since IDDT = 1 is a default value.

example, positioning is at absolute coordinates (0, 1500).[†] The four line segments are drawn, one at a time, in a clockwise direction.

GURSET

```
     CALL GURSET(0, 1500, 102B, IBUF, NBYTE, MBYTE)
```

moves the beam to absolute coordinates (0, 1500). Until there is another GURSET call, all coordinates specified will be relative to absolute (0, 1500).

The third parameter of the call is ICODE and is in the binary form s000fbb[††] where

s	= 0	Disable item's sensitivity to light-pen strike
	= 1	Enable item's sensitivity to light-pen strike
f	= 0	Don't blink item when it is displayed
	= 1	Blink item when it is displayed
bb	= 01	Display item with beam at low intensity
	= 10	Display item with beam at medium intensity
	= 11	Display item with beam at high intensity

The ICODE of CALL GURSET has the bit pattern 1000010 = 102B. It defines the display items as a figure which is sensitive to a light-pen strike, does not blink, and which is displayed at medium beam intensity. If ICODE were 106B, the diamond would blink as the bit pattern would be 1000110. An item with ICODE = 7 would not be sensitive to a light-pen strike, would blink, and would be displayed at high intensity.

The fourth parameter of GURSET is IBUF. IBUF is a description buffer for the display item. The parameter must be dimensioned. Since each byte is 12 bits long, IBUF is usually dimensioned at 64 (60-bit words).

- NBYTE is the number of bytes currently in IBUF. It is updated by the call. It must be initialized to zero at the beginning of a subroutine or program.
- If NBYTE is referenced by later graphics routines, it must be spelled correctly.
- MBYTE is the maximum number of bytes the programmer allows in IBUF. MBYTE is specified by the programmer and must be less than 310. Because each byte is 12 bits, IBUF is usually dimensioned 64 (60-bit words).

[†] See page 2-12 for information on absolute and relative coordinates.

[††] See page 5-12 for an additional option available under SCOPE 3.3

DESCRIBING LINE SEGMENTS

After the light beam is in position, the line segments which form the perimeter of the traffic sign may be drawn. There are several methods of describing line segments.

METHOD I

The first one calls GUSEGS. The command format is

```
        CALL GUSEGS(0, 1500, 1500, 0, 1, 7777B, IBUF,NBYTE, MBYTE)
```

This indicates that the first segment has a starting point at (0, 1500) and endpoint at (1500, 0). The shortest line segment that can be drawn on the 274 Console is approximately 6 dgus long.

The fifth parameter is IBEAM. If IBEAM = 0, the segment is not displayed. If IBEAM = 1, the segment is displayed according to ISTYLE. If IBEAM = -0, the beam is turned off and left off after the last endpoint coordinates are processed. If IBEAM = -1, the beam is turned on and left on after the last endpoint coordinates are processed.

The sixth parameter is ISTYLE, which determines the appearance of the line.

ISTYLE = 0, -0, or 7777B†	The segment is solid	————————
= 5252B	The segment is dashed	-----------
= 6666B	The segment is broken	— — — — —
= 7272B	The segment has an appearance called center line	—.—.—.—

The following code would complete the description of the diamond.

```
       CALL GUSEGS  (1500, 0, 0, -1500, 1, 7777B, IBUF, NBYTE, MBYTE)
       CALL GUSEGS  (0, -1500, -1500, 0, 1, 7777B, IBUF, NBYTE,
      1       MBYTE)
       CALL GUSEGS  (-1500, 0, 0, 1500, 1, 7777B, IBUF, NBYTE, MBYTE)
```

METHOD II

A second method of describing the diamond-shaped perimeter of the warning sign is the following:

```
        CALL GURSET(0, 1500, 102B, IBUF, NBYTE, MBYTE)
        CALL GUSEGS(0, 1500, 1500, 0, 1, 7777B, IBUF, NBYTE, MBYTE)
        CALL GUSEG(0, -1500, 1)
        CALL GUSEG(-1500, 0, 1)
        CALL GUSEG(0, 1500, 1)
```

†7777B is used by convention for ISTYLE since 0 and -0 have other uses such as terminating parameter strings and specifying grid coordinates.

The x and y coordinates of GUSEG are the first and second parameters, respectively. The third parameter is IBEAM which indicates, in this example, that all the line segments are to be displayed. In this context, GUSEGS identifies the relative coordinates of the starting point of the first segment of the figure since GUSEG identifies only endpoints of line segments.

METHOD III

A third method of describing the line segments forming the diamond employs

CALL GUSEGI (IH1, IV1, ISTYLE, IBUF, NBYTE, MBYTE)

GUSEGI should be preceded by a call to GURSET since IH1 and IV1 are coordinates that are relative to the absolute coordinates specified in the last GURSET. GURSET also initializes an IBUF buffer for a new display item description. A display item may be a single line or a complete hexagon as long as its description is in a single IBUF.

GUSEGI determines the starting point of a figure and controls its appearance with the ISTYLE parameter. The code follows.

```
      CALL GURSET(0,1500,103B,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(0,1500,7777B,IBUF,NBYTE,MBYTE)
      CALL GUSEG(1500,0,1)
      CALL GUSEG(0,-1500,1)
      CALL GUSEG(-1500,0,1)
      CALL GUSEG(0,1500,1)
```

METHOD IV

A fourth method of describing the line segments which form the warning sign's perimeter uses a call to GUSEGA. The code is

```
      IH(1) = 0                     $IV(1) = 1500
      IH(2) = 1500                  $IV(2) = 0
      IH(3) = 0                     $IV(3) = -1500
      IH(4) = -1500                 $IV(4) = 0
      IH(5) = 0                     $IV(5) = 1500
      IBEAM(1) = IBEAM(2) = IBEAM(3) = IBEAM(4) = 1
      IBEAM(5) = -0
      CALL GURSET (0, 1500, 103B, IBUF, NBYTE, MBYTE)
      CALL GUSEGA (IH(1), IV(1), IBEAM(1), 4, 7777B, IBUF, NBYTE,
     1    MBYTE)
```

The general call statement format is

CALL GUSEGA (IH, IV, IBEAM, N, ISTYLE, IBUF, NBYTE, MBYTE)

N is the number of segments to be generated by the GUSEGA call. You will notice that N is one less than the number of values in the IH and IV arrays. (These arrays must be dimensioned).

The IH and IV parameters are the first words of the arrays which contain the horizontal and vertical coordinates. IBEAM is the first word of an array containing the beam control code for each figure segment. IBEAM(5) does not describe a segment; with a value of -0, IBEAM(5) turns the light beam off when the diamond is completed.

DISPLAY LINE SEGMENTS

In order to display line segments described by any of the four methods outlined above, the programmer uses

```
        CALL GIDISP (NCON, IBUF, NBYTE, ISQR)
```

NCON is the number of the console that is to be used for the display. Usually NCON is set at the beginning of the program with a card NCON = 1, or NCON is read from a data card at the beginning of the program with the following code.

```
        READ 1, NCON
      1 FORMAT  (I2)
```

The call statement format for GIDISP is

```
        CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,IDDT,IDDC,IDWA,IDWB)
```

The fourth parameter of GIDISP is IDDAD, which is the associative address in the display buffer and is returned by the call. The programmer may substitute any legal FORTRAN integer name for IDDAD. In this case, ISQR describes the square shape of the sign and replaces IDDAD. The IDDAD parameter is necessary to erase an item from the screen with an IGS erase routine.

The fifth through eighth parameters of GIDISP are optional. The fifth parameter is IDDT. If IDDT = 1, the display item is an ignore item. Since 1 is ADDT's default value, the call to GIDISP may be terminated with a right parameter after the IDDAD. The problem definition indicates that an IDDT = 2 specifies that display item as a single pick item. Chapter 3 covers masks and possible values of IDDT. IDDC is an arbitrary word the programmer assigns to identify a particular display item ($0 \leq \text{IDDC} \leq 2^8 - 1$). Perhaps there are three display items with IDDCs of 2, 22, and 200. If two of them are to be erased, the programmer might use a test:

```
        IF (IDDC.NE.22) CALL GIERAS (ITEM1, ITEM2).
```

IDWA and IDWB are ID information words. Their contents are arbitrary unless the item ID block is used by AETSKR. Often, programmers omit IDWA and IDWB and other optional parameters completely. (See Appendix C for an example of use of IDWA.)

SECOND STEP IN CREATION

The second step in displaying the warning sign is displaying the words WINDING ROAD. When calculating the spacing for the words, allow 24 dgus in the vertical direction and 24 dgus in the horizontal direction for each character. WINDING is 24 times 7 dgus, or 168 dgus long. Position W on the screen at (-84, 770). ROAD is 24 times 4 dgus, or 96 dgus long. Position R at (-48, -770).

The routine used to pack alphanumeric information into a display item description buffer is

```
CALL GUAN (IBCD, NC, IBUF, NBYTE, MBYTE)
```

IBCD is the first word of the array of characters to be displayed; each array word consists of ten characters. There are two methods of filling the IBCD array. The first method uses Hollerith notation, and the second employs ENCODE.

NC is the number of characters the programmer wishes to display. The code to display WINDING will be

```
CALL GURSET(-84,770,2,IBUF,NBYTE,MBYTE)
CALL GUAN(7HWINDING,7,IBUF,NBYTE,MBYTE)
CALL GIDISP(NCON,IBUF,NBYTE,IWIND)
```

ENCODE is not commonly used to convert input from the 274 Console, but it can be used for display of long strings of alphanumeric data.

To display ROAD using ENCODE, the programmer would code

```
  CALL GURSET (-48, -770, 2, IBUF, NBYTE, MBYTE)
  ENCODE (4, 2, IBCD)
2 FORMAT (4HROAD)
  CALL GUAN (IBCD, 4, IBUF, NBYTE, MBYTE)
  CALL GIDISP (NCON, IBUF, NBYTE, IROAD)
```

The first ENCODE parameter indicates the number of characters to be encoded, the second parameter is the number of the format by which information is to be translated, and IBCD is the FORTRAN integer name attached to the character array. An optional list of additional information to be coded may be included.

THIRD STEP IN CREATION

The third step in creation of the traffic sign is displaying the curve. The user can form the curve by connecting two arcs from circles to form an S-shape. If the centers of the circles containing the desired arcs and their endpoints are calculated, the user can use the IGS call GUARCG to describe the arcs which are to appear on the console. The circles or arcs described by GUARCG are drawn in a counterclockwise direction. The user might specify (256, 256) as the starting point of the arc in the top circle whose center is (0, 256). The endpoint is (0, 0). The second arc might have a starting point of (-256, -256), a center of (0, -256) and an endpoint of (0, 0).

The call format is

```
        CALL GUARCG (KSHOW, IHC, IVC, IH1, IV1, IH2, IV2,
       1ISTYLE, IBUF, NBYTE, MBYTE)
```

KSHOW indicates the number of arc segments generated by this call where all the segments have a common circle center. IHC and IVC are coordinates of the common center of the arcs. IH1 and IV1 are the starting point coordinates of the counterclockwise arc, and IH2 and IV2 are the coordinates of the endpoint of the arc. To display the S-shaped curve, position the light beam at the starting point of the upper arc and describe the arc. Reposition the beam to the starting point of the lower arc, describe the lower arc, and then display both arcs.

```
       CALL GURSET(256,256,102B,IBUF,NBYTE,MBYTE)
       CALL GUARCG(1,0,256,256,256,0,0,7777B,IBUF,NBYTE,MBYTE)
       CALL GURSET(-256,-256,102B,IBUF,NBYTE,MBYTE)
       CALL GUARCG(1,0,-256,-256,-256,0,0,7777B,IBUF,NBYTE,MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,IS)
```

By putting the three sections of code together with some control cards (which are discussed later), the operator can display a traffic caution sign warning WINDING ROAD.

```
       ISLO,P17,T10000,CM40000.
       RUN(S)
       LGO.
       AEFILE.
       7-8-9
             OVERLAY(SOURCE,0,0)
             PROGRAM CREATE
             CALL MAIN
             END
             OVERLAY(1,0)
             PROGRAM JSLOW
             DIMENSION  IBUF(64),IBCD(10),IDDAD(10)
             NCON=1
             NBYTE=0
             MBYTE=310
```

```
C      ATTACH CONSOLE AND CLEAR BUFFERS
       CALL GICNJB(NCON)
C      DISPLAY DIAMOND SHAPED PERIMETER
       CALL GURSET(0,1500,102B,IBUF,NBYTE,MBYTE)
       CALL GUSEGS(0,1500,1500,0,1,7777B,IBUF,NBYTE,MBYTE)
       CALL GUSEG(0,-1500,1)
       CALL GUSEG(-1500,0,1)
       CALL GUSEG(0,1500,1)
       CALL GIDISP(NCON,IBUF,NBYTE,ISQR)
C      DISPLAY *WINDING ROAD*
       CALL GURSET(-84,770,2,IBUF,NBYTE,MBYTE)
       CALL GUAN(7HWINDING,7,IBUF,NBYTE,MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,IWIND)
       CALL GURSET (-48,-770,2,IBUF,NBYTE,MBYTE)
       CALL GUAN(4HROAD,4,IBUF,NBYTE,MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,IROAD)
C      DISPLAY CURVE
       CALL GURSET(256,256,102B,IBUF,NBYTE,MBYTE)
       CALL GUARCG(1,0,256,256,256,0,0,7777B,IBUF,NBYTE,MBYTE)
       CALL GURSET(-256,-256,102B,IBUF,NBYTE,MBYTE)
       CALL GUARCG(1,0,-256,-256,-256,0,0,7777B,IBUF,NBYTE,MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,IS)
       END
7-8-9
OBJECT
SOURCE
6-7-8-9
ISLO,P17,T10000,CM40000.
RUN(S)
COMMON,OBJECT.
LGO.
RELEASE,OBJECT.
EXIT.
RELEASE,OBJECT.
7-8-9
       OVERLAY(SOURCE,0,0)
       PROGRAM CREATE
       CALL MAIN
       END
7-8-9
OBJECT
6-7-8-9
```

If a version of IGS earlier than the 3.2 version is issued, then each CALL GIDISP must be followed with a card that reads NBYTE = 0. The user should now be able to display an x- and y-axis with name labels and a number label at the origin; he might wish to draw an arc or curve on the x-y grid.

COORDINATES—ABSOLUTE AND RELATIVE

Absolute coordinates always refer to x-y coordinates on the 274 Console. The origin (0, 0) is always at the center of the screen. Only five IGS calls ever specify absolute coordinates: GURSET, GICOPY (covered in Chapter 3), GIMOVE(covered in A-6), GITCON, and GITCOF.

```
        CALL GURSET (100, 0, 103B, IBUF, NBYTE, MBYTE)
```

always refers to point A on the diagram in Figure 2-4, when A is 100 dgus from the center on the x-axis.

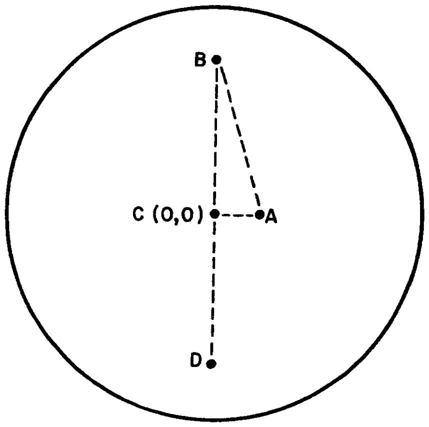


Figure 2-4. Short Line Segment Diagram

EXAMPLE 1.

Assume that the following code appears in an application program:

```
      CALL GURSET(0,0,103B,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(0,0,100,0,1,7777B,IBUF,NBYTE,MBYTE)
      IF(N.GT.3)GOTO 100
      CALL GURSET(0,1800,103B,IBUF,NBYTE,MBYTE)
```

The second call to GURSET does not depend on the first call to GURSET. The first call moves the light beam to point C. The second moves the light beam to point B at absolute coordinates (0, 1800), where B is on the y-axis 1800 dgus from the origin, at point C.

EXAMPLE 2.

The routines GUSEGS, GUSEG, GUSEGI, and GUSEGA specify relative coordinates. They are preceded by a call to GURSET.

```
CALL GURSET(0,1800,103B,IBUF,NBYTE,MBYTE)
CALL GUSEGS(0,1800,0,1806,1,7777B,IBUF,NBYTE,MBYTE)
```

describes a short line segment which appears as a point at B on the diagram with C the relative origin as well as the absolute origin.

GUSEGS, GUSEG, GUSEGI, and GUSEGA use the first two parameters to name the point where the beam is located after the call to GURSET. In the case where the first two parameters are the same as the first two parameters of GURSET, the relative and absolute coordinates of the points later specified will be the same. If the pairs of parameters differ, a relative origin has been defined. Any point on the screen may thus be defined as a relative origin.

EXAMPLE 3.

```
CALL GURSET(0,1800,103B,IBUF,NBYTE,MBYTE)
CALL GUSEGS(0,0,0,6,1,7777B,IBUF,NBYTE,MBYTE)
```

describes the same short line segment (as in Example 2) which appears as a point at B on the diagram. This time, GUSEGS redefines the position of the light beam as relative (0, 0). C is at relative (0, -1800) and A is at relative (100, -1800). D is on the y-axis 1200 dgus from the center of the screen. In this case, D is at relative (0, -3000) but D cannot be referenced by a call to GUSEGS, GUSEG, GUSEGI, or GUSEGA with the current relative origin since the absolute value of the relative coordinates of these calls must be less than 2048 dgus. Be sure to watch for this as it is a common error to attempt to reference coordinates with absolute values greater than 2048 dgus.

EXAMPLE 4.

```
CALL GURSET(0,1800,103B,IBUF,NBYTE,MBYTE)
CALL GUSEGI(100,+2000,7777B,IBUF,NBYTE,MBYTE)
CALL GUSEG(100,+2006,1)
```

GUSEGI redefines the coordinates of point B as relative (100, +2000) with relative origin at (-100, -200). It describes a very short segment (a point) drawn at B. C has relative coordinates (100, 200), A has relative coordinates (200, 200), and D has relative coordinates (100, -1000).

The subroutines CIRCLE and LINE on page A-16 and A-17 are examples of a circle and square whose coordinates become relative to the GURSET coordinates when they are displayed. PROGRAM MOVE and PROGRAM COPY on pages A-42 and A-43 are examples of the use of absolute and relative coordinates.

REVIEW QUESTIONS

1. Each alphanumeric character with spacing is displayed in a square of what dimensions?

   a. 32 dgu × 32 dgu
   b. 24 dgu × 24 dgu
   c. 10 dgu × 10 dgu

2. On the 274 Console, there are approximately how many digigraphic units (dgus) per inch?

   a. 200_8
   b. 200
   c. 256

3. A horizontal line 1024 dgus long would be completely displayed if the left-most endpoint were positioned at:

   a. (256, 1024)
   b. (512, 1536)
   c. (-1792, 1280)

4. If the programmer specifies absolute coordinates (1024, 1024) as the relative origin (0, 0) for a display item, he can reference the absolute coordinates:

   a. (-1280, 512)
   b. (1024, -1280)
   c. (256, 768)

5. The width of a line drawn on the 274 Console at 50% intensity is approximately:

   a. 5 dgus
   b. 1 dgu
   c. 30 dgus

6. The calls

```
CALL GURSET (0, 0, 2, IBUF, NBYTE, MBYTE)
CALL GUSEGS (400, 200, 0, 1, 0, IBUF, NBYTE, MBYTE)
CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)
```

and

```
CALL GURSET (400, 200, 2, IBUF, NBYTE, MBYTE)
CALL GUSEGS (400, 200, 0, 1, 0, IBUF, NBYTE, MBYTE)
CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)
```

   a. Generate a line in the same location

b. Generate a line in different locations

c. Both set the relative origin at (0, 0)

7. The shortest line segment that can be drawn with a GUSEGS, GUSEG, or GUSEGA is:

a. 0 dgu (a point)

b. 1 dgu (a point)

c. 6 dgu (a point)

The MBYTE that is too great is:

a. 310

b. 290

c. 400

9. The following statement about NBYTE is false:

a. NBYTE is reset to zero after a GIDISP call.

b. An NBYTE EXCEEDS MBYTE error message can indicate NBYTE was not reset by the programmer or software when a new IBUF was started.

c. Misspelling of NBYTE does not constitute a recognized error.

10. GURSET –

a. Drives the light beam to absolute screen coordinates given by the call.

b. Moves the beam to the relative coordinates given by the call.

c. Requires all but the sixth parameter in its parameter list.

Choose answers from a through g to describe the item whose initial conditions are established by the call to GURSET:

11. CALL GURSET (100, 100, 103B, IC1RC, NBYTE, MBYTE)

12. CALL GURSET (-200, 0, 2, ISQR, NBYTE, 300)

13. CALL GURSET (IH, IV, 107B, IBUF, NBYTE, MBYTE)

a. Disable light-pen sensitivity of item

b. Enable light-pen sensitivity of item

c. Don't blink item when displayed

d. Blink item when displayed

e. Display item at low intensity

f. Display item at medium intensity

g. Display item at high intensity

14. Which of the following GIDISP calls are illegal?

    a. CALL GIDISP (NCON, IBUF, NBYTE, ISQR)

    b. CALL GIDISP (NCON, IBUF, NBYTE, RET)

    c. CALL GIDISP (NCON, IRES, NBYTE, IDDAD, 1, 2000)

    d. CALL GIDISP (NCON, IBUF, NBYTE, IDDAD1, 1, 33, 0, 0)

15. Which of the following GUAN calls are illegal?

    a. CALL GUAN (IBCD, 10, IBUF, NBYTE, MBYTE)

    b. CALL GUAN (5HLABEL, 5, PRES)

    c. CALL GUAN (IREMARK, 6, IBUF, NBYTE, MBYTE)

    d. CALL GUAN (IBCD, 7, IBUF, NBYTE, 300)

16. What does the following code do if standard masks are set?

```
CALL GURSET (-500, 500, 103B, ILBL, NBYTE, MBYTE)
CALL GUAN (5HTITLE, 5, ILBL, NBYTE, MBYTE)
CALL GIDISP (NCON, ILBL, NBYTE, LABL, 2, 22)
```

    a. Displays TITLE at medium intensity as a string pick item at (-500, 500).

    b. Displays TITLE as a single pick item at high intensity at (-500, 500).

    c. Displays TITLE as a single pick item that is light-pen sensitive at high intensity with associative address ILBL.

17. CALL GUSEGI (-300, 0, 7777B, IBUF, NBYTE, MBYTE)

    a. Initializes a display item at absolute coordinates (-300, 0).

    b. Specifies that a figure will be drawn with solid lines and begins at relative coordinates (-300, 0).

    c. Initializes a dashed line display item with starting point at (-300, 0) relative to the last call to GURSET.

18. CALL GUSEG (-300, 1000, 1)

   a. Is used to specify the endpoint of a line segment not to be displayed, whose starting point is specified by a CALL GUSEGI or a CALL GUSEG.

   b. Specifies (-300, 1000) relative to the last CALL GUSEGI as the endpoint of a line segment.

   c. Specifies absolute (-300, 1000) as a line segment endpoint following CALL GURSET (0, 0, 102B, IBUF, NBYTE, MBYTE) and GUSEGI (-300, 300, 0, IBUF, NBYTE, MBYTE).

19. CALL GUSEGS (20, 100, 1000, 700, 1, 6666B, IBUF, NBYTE, MBYTE) –

   a. Generates a description of a broken line segment with relative endpoints at (20, 100), (700, 1).

   b. Describes a dashed line segment with relative endpoints at (20, 100) and (1000, 700).

   c. Describes a broken line segment with relative endpoints at (1000, 700) and (20, 100).

20. CALL GUSEGA follows CALL GURSET (-50, 50, 103B, NBYTE, MBYTE). For CALL GUSEGA (IH(1), IV(1), J(1), 3, 7777B, IBUF, NBYTE, MBYTE) to generate a triangle with a vertex at absolute (-50, 50), the values of IH(1) and IV(1) cannot be:

   a. IH(1)=-50, IV(1)=50/IH(2)=50, IV(2)=50/IH(3)=0, IV(3)=50/IH(4)=-50, IV(4)=50

   b. IH(1)=-50, IV(1)=50/IH(2)=50, IV(2)=50/IH(3)=0, IV(3)=-50

   c. IH(1)=50, IV(1)=50/IH(2)=0, IV(2)=-50/IH(3)=-50, IV(3)=50/IH(4)=50, IV(4)=50

   When J(1) = J(2) = J(3) = 1.

21. The CALL GUARCG (3, 100, 100, IH(1), IV(1), JH(1), JV(1), 7777B, IBUF, NBYTE, MBYTE) will generate a description of:

   a. 3 arcs when IH(1)=0, IV(1)=0/JH(1)=100, JV(1)=-100/IH(2)=200, IV(2)=0/JH(2)=100, JV(2)=100.

   b. 3 arcs when IH(1)=0, IV(1)=0/IH(2)=200, IV(2)=0/IH(3)=100, IV(3)=100/JH(1)=100, JV(1)=100/JH(2)=100, JV(2)=100/JH(3)=0, JV(3)=0.

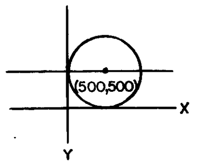
   c. 3 arcs when KSHOW = 3 and IH(I), IV(I), JH(I), JV(I) are 3 by 3 arrays.

CODING PROBLEMS

Assume masks are set such that

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker
MBYTE	=	310
NCON	=	1

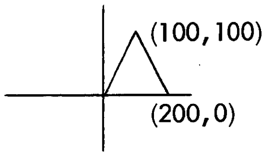
1. Display a solid line circle with radius of 500 dgus and center (500, 500) as an ignore item that blinks.



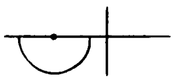
2. Display your first and last names on the console centering your names at (0, 0).

3. Use GUSEGS to display a horizontal dashed line 1500 dgus long on the screen with left endpoint at (-1500, -1000).

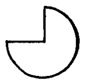
4. Use GURSET, GUSEGI and GUSEG to display an isosceles triangle with one vertex at (100, 100).



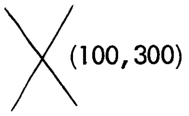
5. Display an enclosed semicircle with center (-200, 0) and radius 100 dgus.



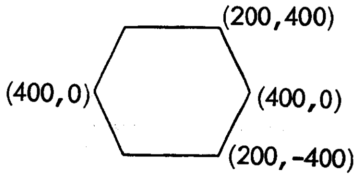
6. Use three two-line segments and an arc to display the first, fourth and third quadrants of a circle with lines marking the y-axis boundary of the enclosed portion of quadrant I and marking the x-axis boundary of the enclosed portion of quadrant III. Make it an ignore item with center at (-200, 0) and radius 100 dgus.



7. Use GUSEGI and GUSEG to display an X 50 dgus wide and 70 dgus high with center at point (100, 300) on the screen. Label the center point.



8. Suppose we wish to draw the hexagon at the right. There is an array of horizontal coordinates and an array of vertical coordinates listed in clockwise order such that



```
IH(1) = -200          IV(1) =  400
IH(2) =  200          IV(2) =  400
IH(3) =  400          IV(3) =  0
IH(4) =  200          IV(4) = -400
IH(5) = -200          IV(5) = -400
IH(6) = -400          IV(6) =  0
IH(7) = -200          IV(7) =  400
IBEAM(1)=IBEAM(2)=IBEAM(3)=IBEAM(4)=IBEAM(5)
=IBEAM(6)=1
```

Use GURSET and GUSEGA to describe and display the figure.

DRIFTING CIRCLE 3

One feature of the Interactive Graphics System is that the user can interact with the computer by using a light-pen or, on some systems, a keyboard. In order to illustrate some of the capabilities of IGS when a light-pen is used, the manual presents the Drifting Circle[†] example (Figure 3-1). In this example, there is a display of a console title DRIFTING CIRCLE, and a display of a circle as a macro. The programmer attaches a tracking cross to the circle, moves the circle to the right side of the screen, and copies the circle on the left side of the screen. He then erases the entire screen by picking a REPEAT button display item.

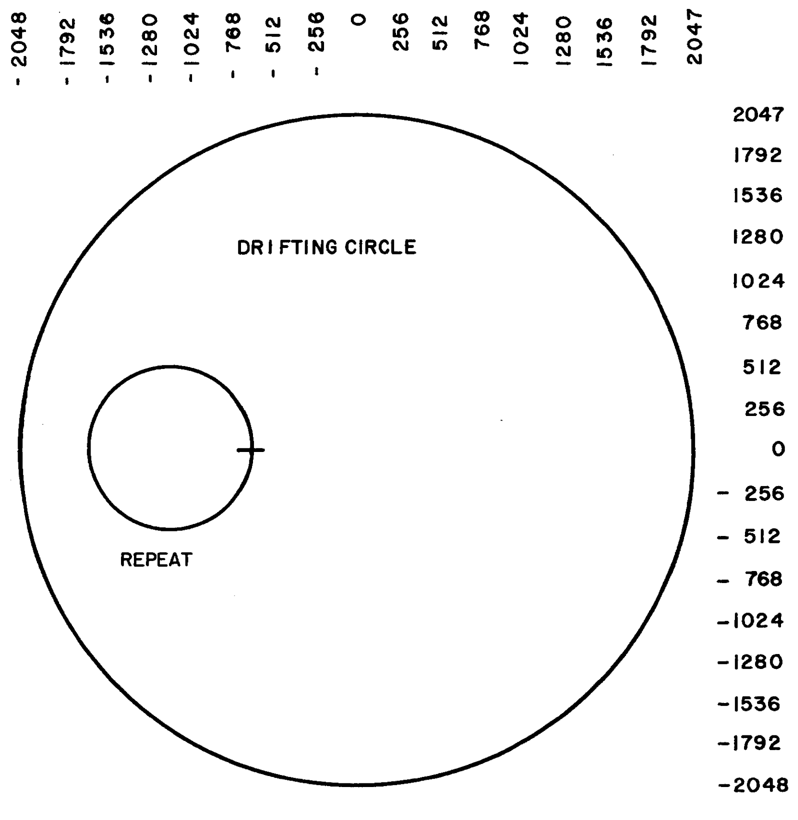


Figure 3-1. Drifting Circle

[†]Appendix A includes a different approach to the Drifting Circle problem.

DISPLAYING THE TITLE

The console title is 15 times 24 dgus, or 360 dgus long. Position the lower left-hand corner of the D at (-180, 1000). The code to display the title using ENCODE is

```
          CALL GURSET(-180,1000,102B,IBUF,NBYTE,MBYTE)
          ENCODE(15,1,IBCD)
      1   FORMAT(15HDRIFTING CIRCLE)
          CALL GUAN(IBCD,15,IBUF,NBYTE,MBYTE)
          CALL GIDISP(NCON,IBUF,NBYTE,ILBL)
```

or

```
          DATA IBCD/15HDRIFTING CIRCLE/
              :        :         :
          CALL GURSET(-180,1000,102B,IBUF,NBYTE,MBYTE)
          CALL GUAN(IBCD,15,IBUF,NBYTE,MBYTE)
          CALL GIDISP(NCON,IBUF,NBYTE,ILBL)
```

DISPLAYING THE CIRCLE

Use GUARCG to display the circle. To display a circle with center at (-1024, 0) and a radius of 512 dgus, the routines GURSET, GUARCG, and GIDISP may be used.

The problem definition calls for the circle to be displayed as a macro. The term macro, when related to IGS, refers to an item which can be displayed in many places on the screen without repetition of the description of the item. The use of macros conserves display buffer memory.

CALL GUARCG stores the description of the circle in IBUF. The programmer who follows CALL GUARCG with CALL GIMAC(NCON, IBUF, NBYTE, MAD) stores the display item description as a display macro.

MAD is the associative address in the display buffer of the new macro which is returned as a result of the CALL. MCIRC or any other legal FORTRAN integer name could be substituted for MAD in the parameter string. When the programmer refers to this macro in later calls, he should reference it by the address name that has been selected, in this case MAD.

A macro can be displayed as an ignore item (IDDT = 1), a single pick item (IDDT = 2), or some other type of item. In this case, the following code makes the circle a macro and displays it as a button pick item (IDDT = 8), with center at (-1024, 0) and starting point at (-512, 0). The display item code (IDDC) has been set to 33. A light pen pick of a sensitive button pick item causes a program branch.

```
      CALL GUARCG(1,-1024,0,-512,0,-512,0,7777B,IBUF,NBYTE,
     1  MBYTE)
      CALL GIMAC(NCON,IBUF,NBYTE,MCIRC)
C     CREATE CIRCLE MACRO
      CALL GURSET(-512,0,102B,IBUF,NBYTE,MBYTE)
      CALL GUMACG(MCIRC,1,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ITEM1,8,33)
```

If a GURSET had been used before GUARCG, the macro could not be moved. The CALL GUMACG (MAD1, L, IBUF, NBYTE, MBYTE) must be used with GIDISP to display the circle macro. MAD1 is the first word of an array containing parameters returned by earlier calls to GIMAC. If the user wishes to display four concentric circles as one display item and each circle is a macro, L = 4 as long as the concentric circles have consecutive MAD parameters. L = 1 when one macro is displayed as one display item.

MOVING A MACRO

The macro circle may also drift. One method involves attaching a tracking cross to the macro. A tracking cross is a software display item —┼— which always exists somewhere on the console screen. If the programmer desires, he can display the cross at a point (such as 1792, 1792) which is off the circular 274 screen and, therefore, invisible to the user.

When attaching the tracking cross, a call to GITCON should turn on the tracking cross at the initial or starting position of a display item. To turn on the tracking cross at this circle's starting point (-512, 0), the call

```
CALL GITCON(NCON, -512, 0)
```

is used.

The last two parameters are the x and y screen coordinates where the center of the tracking cross is to appear. There is a key on the light-pen which can be used in various picking situations. In this case, the pen is used to pick the tracking cross at location (-512, 0). The cross is then automatically attached to the pen and moves with the light-pen as the pen is moved across the screen.

Attach the circle macro to the tracking cross with

```
CALL GITMMV(NCON, MCIRC)
```

where MCIRC is the associative address of the circle macro that was specified in CALL GUMACG. If the circle was only a display item, not a macro, the operator would use

```
CALL GITIMV(NCON, IDDAD)
```

where IDDAD is the associative address of the display item which will be attached to the tracking cross.

The light-pen key is activated by using

CALL GILPKY(NCON, 8, 7)

IDDT = 8, which indicates the release of the key is queued as a button pick. The code (IDDC) associated with the button pick is 7. The fourth and fifth parameters of GILPKY are IDWA and IDWB. They are optional unless the AE routines are used for this button.

In the Drifting Circle example, the user simply picks the tracking cross with the light-pen and, with the light-pen key depressed, tows the circle to the right side of the screen. Suppose he stops the light-pen and releases the key at approximately (600, 0) on the screen (Figure 3-2). The release of the light-pen key queues a button pick. This pick has IDDC = 7. Display of the circle at its new location is done only after towing is complete. A button pick test can be inserted here. Another test verifies that the tracking cross was in fact moved. In this example, the circle can be moved three different times.

GIBUT TESTS

To simply move the circle ITEM with starting point (-512, 0) to a position where the starting point is (1000, 600), the call is

CALL GIMOVE (1000, 600, 102B, ITEM 1, 8, 33)

where the general call is

CALL GIMOVE(IH, IV, ICODE, IDDAD, IDDT, IDDC, IDWA, IDWB)

GICLR clears all ID blocks from the queues so that multiple button picks do not cause confusion. The GIBUT routine fetches the first sequential button pick item from the queue and returns the last six parameters in its parameter string.

The first parameter, IR, is zero if the processing is to wait until a button pick type ID block is queued. IR = 1 if a return to the calling task is expected immediately.

```
      K=0
201   CALL GICLR(NCON)
      CALL GIBUT(0,NCON,IDDT,IDDC,IDWA,IDWB,IH,IV)
      K=K+1
      IF(K.EQ.3) GO TO 202
      IF(IDDC.NE.7) GO TO 201
      IF(IH.NE.-512.AND.IV.NE.0) GO TO 201
```

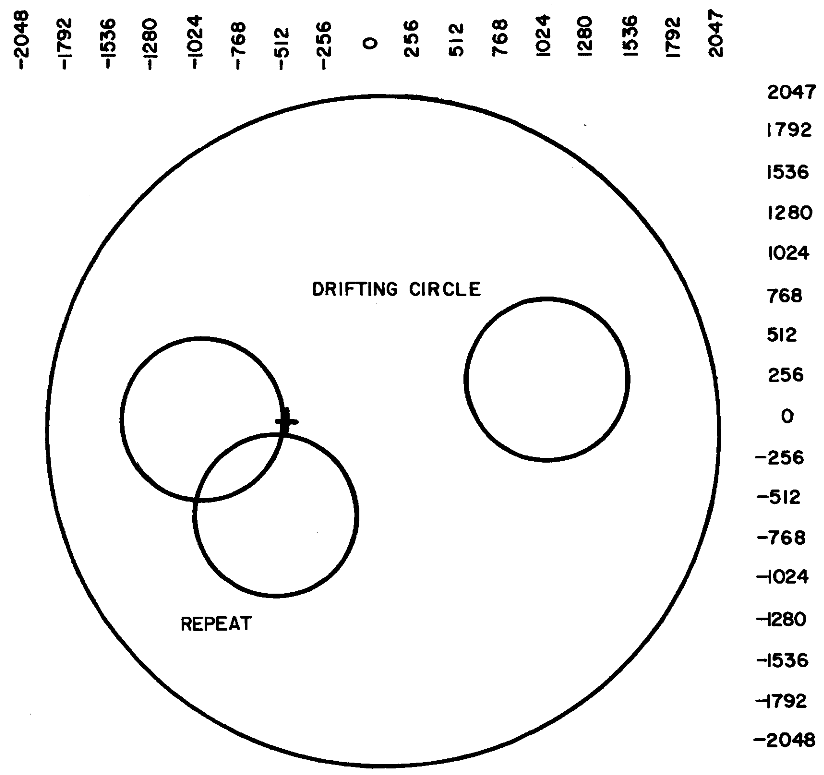


Figure 3-2. Towing the Drifting Circle

DISPLAYING THE MOVED MACRO IN ITS NEW LOCATION

IH and IV are the coordinates of the light-pen pick which caused the block to be queued. They are returned as a result of the call. If GILPKY created the ID block, the exact tracking cross coordinates at the time of interrupt are returned. Because the tracking cross coordinates at the time of pick do not always coincide with the starting point of the macro, the approximate coordinates for the starting point of the macro to be displayed appear as an ignore item after the move associated with the last button pick. CALL GITCOF(NCON, IH, IV) would also return coordinates of the tracking cross.

```
CALL GURSET(IH,IV,102B,IBUF,NBYTE,MBYTE)
CALL GUMACG(MCIRC,1,IBUF,NBYTE,MBYTE)
CALL GIDISP(NCON,IBUF,NBYTE,ITEM2,1)
```

STATUS CHANGE

To change the status of the light-pen key from a button to an ignore item, the operator would call GILPKY.

```
        CALL GILPKY(NCON)
```

DETACH CROSS

Though it is not necessary to detach the tracking cross, this is an opportunity to demonstrate the method of detaching a tracking cross from a macro. When the tracking cross is detached, it can be moved away from the macro without moving the macro.

```
        CALL GITMMV(NCON, 0)
```

COPY CIRCLE

The original circle on the left of the screen is displayed as well as the circle on the right-hand side. In order to clear part of the screen for a copy of the moved circle, ITEM2, the display item ITEM1 is erased.

```
        CALL GIERAS(ITEM1)
```

The call statement format is

        CALL GIERAS(IDDAD$_1$, IDDAD$_2$, . . . IDDAD$_n$)

where IDDAD$_n$ is the associative address of the item to be erased.

In order to copy the display item, ITEM2, with starting point at (-300, -100), GICOPY is called.

```
        CALL GICOPY(ITEM2,NCON, -300, -100, 106B, ITEM3, 1)
```

This call duplicates the circle ITEM2, assigns a new ID block and reset sequence to the copy ITEM3, and displays the copy at (-300, -100).

The first parameter is IDDADI, the associative address of the item to be duplicated. The fifth parameter is ISTYLE which specifies a blinking circle displayed at medium intensity. The sixth parameter is the IDDAD of the new item. The seventh parameter is IDDT.

ERASE CONSOLE

The last task is erasing the console by picking a REPEAT button. The code for this task is included in the listing of the program.

CONSOLE INITIALIZATION

At the beginning of the listing, with the dimension and initialization code, there is a new IGS call. CALL GICNJB(NCON) attaches the console with the number equal to NCON and clears tables and masks from previous programs.

```
JDRIFT,P17,T10000,CM40000.
RUN(S)
LGO.
AEFILE.
7/8/9
      OVERLAY(SOURCE,0,0)
      PROGRAM CREATE
      CALL MAIN
      END
      OVERLAY(1,0)
      PROGRAM DRIFT
      DIMENSION IBUF(64), IBCD(10),ILBL(2)
      NCON=1
C     ATTACH CONSOLE
      CALL GICNJB(NCON)
      MBYTE=310
      NBYTE=0
C     SET MASKS
      DO 100 I=2,5
      N=2**(I-1)
100   CALL GIMASK(NCON,-0,N,N,)
C     DISPLAY REPEAT BUTTON
101   CALL GURSET(-1200,-1200,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(6HREPEAT,6,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ICLR,B,44)
C     DISPLAY TITLE
      CALL GURSET(-180,1000,102B,IBUF,NBYTE,MBYTE)
      ENCODE(15,1,IBCD)
1     FORMAT(15HDRIFTING CIRCLE)
      CALL GUAN(IBCD,15,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ILBL,2)
      CALL GUARCG(1,-1024,0,-512,0,-512,0,7777B,IBUF,NBYTE,MBYTE)
      CALL GIMAC(NCON,IBUF,NBYTE,MCIRC)
C     CREATE CIRCLE MACRO
      CALL GURSET(-512,0,102B,IBUF,NBYTE,MBYTE)
      CALL GUMACG(MCIRC,1,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ITEM1)
C     TURN TRACKING CROSS ON
      CALL GITCON(NCON,-512,0)
C     ATTACH MACRO TO TRACKING CROSS
      CALL GITMMV(NCON,MCIRC)
C     ACTIVATE LIGHT PEN KEY
      CALL GILPKY(NCON,8,7)
      K=0
202   K=K+1
201   CALL GICLR(NCON)
      CALL GIBUT(0,NCON,IDDT,IDDC,IDWA,IDWB,IH,IV)
      IF(IDDC.EQ.1)GO TO 210
      IF(IDDC.GE.2.AND.IDDC.LT.7)GO TO 203
      IF(K.LT.3)GO TO 202
```

```
        IF(IDDC.NE.7)GO TO 201
C     DISPLAY CIRCLE IN NEW LOCATION
        CALL GURSET(IH,IV,102B,IBUF,NBYTE,MBYTE)
        CALL GUMACG(MCIRC,1,IBUF,NBYTE,MBYTE)
        CALL GIDISP(NCON,IBUF,NBYTE,ITEM2)
        CALL GILPKY(NCON)
        CALL GITMMV(NCON,0)
        CALL GIERAS(ITEM1)
C       DISPLAY A COPY OF CIRCLE
        CALL GICOPY(ITEM2,NCON,-300,-100,106B,ITEM3)
C       ERASE CONSOLE
200     CALL GIBUT(0,NCON,IDUM,IDX)
        IF(IDX.NE.44)GO TO 200
        CALL GIERAS(ILBL,ICLR,ITEM2,ITEM3)
        GO TO 101
        STOP
        END
7/8/9
OBJECT
SOURCE
6/7/8/9
```

CLEAR MACROS

If the programmer needed to clear macros from the console controller's display buffer so he could use the area for other macros, he would use CALL GIMACE(MAD$_1$, . . . MAD$_n$). MAD$_n$ is the buffer address of the macro to be erased. The call which would erase the circle macro in the Drifting Circle example is

CALL GIMACE(MCIRC)

See Appendix A example, ERASE A MACRO.

TERMINATE JOB

A CALL GIABRT(NCON, IBCD, NC) card could have been used in the Drifting Circle example to voluntarily terminate the job at any point during execution. GIABRT displays an abort message which is stored in IBCD. NC is the number of characters in the message. To display the message JOB COMPLETE, the code would be

CALL GIABRT(NCON, 12HJOB COMPLETE, 12)

GIABRT also enables EXIT control cards and dumps to be processed. Whenever the user signs off or aborts, he should display a message. He can use GURSET, GUAN and GIDISP for a message preceding a normal termination with GUAN and GIDISP for a message preceding a normal termination with GICNRL.

SET MASKS

The programmer must set his own masks at the beginning of each program. Three lines of code following the comment card SET MASKS specify the mask IDDT values for the Drifting Circle example. Using a DO loop and CALL GIMASK(NCON, IDDTC, IDDTS, IMASK), the programmer sets masks at the beginning of this program.

IDDTC is the value of the bit pattern to be cleared from the specified pick processing masks. In the Drifting Circle example, all masks had been set to zero because CALL GICNJB(NCON) preceded the mask setting loop. GICNJB attaches a console or aborts a calling job if the console number, NCON, is not valid or if the console is not available. GICNJB also clears tables and previously set masks.

The third parameter of GIMASK is IDDTS which is the value of the bit pattern to be set in the masks specified.

IMASK may be any one or any combination of the following:

=1	Ignore mask
=2	Single pick mask
=4	String pick mask
=8	Button pick mask
=16	Marker mask

The ignore mask for IDDT need not be set. A pickable item is displayed as a single pick, string pick, or button pick item.

Mask Examples:

- To initially set a mask for a button where an IDDT of 8 indicates a button pick, the operator must set the mask in the button pick mask with IMASK = 8.

  ```
  CALL GIMASK(NCON, 0, 8, 8)
  ```

- The mask in the button pick mask area need not be set to 8. If the operator initially wishes an IDDT = 2 to indicate a button, use

  ```
  CALL GIMASK(NCON, 0, 2, 8)
  ```

  The user may wish to display a menu of button pick items with IDDT = 2 and may subsequently want to change these items to buttons with IDDT = 8 so he can display another menu of single pick items with IDDT = 2 to be associated with these buttons.

- To change a button pick mask that is currently set to $\boxed{000|010}$, to a button mask where an IDDT of 8 indicates a button pick, use

  ```
  CALL GIMASK(NCON, 2, 8, 8)
  ```

  which simultaneously clears and sets the button pick.

- To initially set masks so a single pick will be marked (i.e., will change blinking status when it is picked), use

  ```
  CALL GIMASK(NCON, 0, 2, 2)
  CALL GIMASK(NCON, 0, 16, 16)
  ```

  An IDDT = 18 or IDDT = 16 + 2 or IDDT = 22B would indicate a marked single pick item with these mask settings. To simultaneously set 2 in the single pick mask and the marker mask, use

  ```
  CALL GIMASK(NCON, 0, 2, 2+16)
  ```

The original blinking status of an item is reversed when the marked item is queued. When the item is fetched through a GIBUT or AETSKR, the item resumes its original blinking status.

If the user plans to display an item so that it is a single pick, string pick, or button pick item, an ICODE such as 103B (1000011, enable item's sensitivity to light-pen strike) and an IDDT of 2, 4, 8 or a combination with 16 is recommended. Suppose a button is displayed with an ICODE of 103B (not blinking) and it is desirable to have it blink when picked with the light-pen. The IDDT of the button will be 8 + 16 = 24. In the special case where the button is to be a prime button which can be queued as both a button and a string pick or a button and a single pick, the IDDT might be 4 + 8 + 16 = 28 or 2 + 8 + 16 = 26. These numbers can be expressed in octal as 34B or as a sum such as 12 + 8 in the parameter string as well as in decimal.

REVIEW QUESTIONS

1. Light buttons –

   a. Are a displayed letter, symbol, or word to be picked with the light-pen.

   b. Are circles of light that appear on the console whenever a light-pen pick is to be made.

   c. Are always the buttons on the function keyboard or alphanumeric keyboard.

2. Which of the following parameters would not be used, by convention, to identify an item you wished to erase with GIMACE or GIERAS ?

   a. IDDAD

   b. MAD

   c. IDWA

3. GIMASK requires the following parameters:

   a. NCON, IDDTC, IDDTS

   b. NCON, IDDTC, IDDTS, IMASK

   c. NCON, IDDTC, IMASK

4. CALL GIMASK (NCON, 0, 30B, 30B) indicates:

   a. A button mask and a marker mask are set.

   b. An item is displayed at high intensity and blinks.

   c. A single pick, a string pick mask, a button pick mask, and a marker mask are set.

5. In order to set a single pick mask and a marker mask, the call would not be:

   a. CALL GIMASK (NCON, 0, 188, 188)

   b. CALL GIMASK (NCON, 0, 22B, 22B)

   c. CALL GIMASK (NCON, 0, 16+2, 16+2)

6. What does the following code do?

```
CALL GURSET(-400, 600, 106B, MSG, NBYTE, MBYTE)
ENCODE(33, 1, MESAGE)
```

```
1   FORMAT (33HUSE FONT TO INPUT PARAMETER NAMES)
    CALL GUAN (MESAGE, 33, MSG,NBYTE, MBYTE)
    CALL GIDISP (NCON, MSG, NBYTE, MERASE, 1)
```

a. Displays a 33-character message as a blinking single pick item at (-400, 600).

b. Displays an ignore item message 33 characters long at medium intensity, that can be erased with CALL GIERAS (MERASE).

c. Displays the message in MESAGE as a blinking ignore item, with IDDAD being MSG.

7. Which of the following does not display STATEMENT at (-300, 200) successfully?

a.
```
    CALL GURSET (-300, 200, 102B, ISTATE, NBYTE, MBYTE)
    IBCD (1) = 9HSTATEMENT
    CALL GUAN (IBCD(1), 9, ISTATE, NBYTE, MBYTE)
    CALL GIDISP (NCON, ISTATE, NBYTE, JSTMT, 8, 22)
```

b.
```
    ENCODE (9, 1, IBCD(1))
  1 FORMAT(9HSTATEMENT)
    CALL GURSET (-300, 200, 102B, ISTATE, NBYTE, MBYTE)
    CALL GUAN (IBCD(1), 9, ISTATE, NBYTE, MBYTE)
    CALL GIDISP (NCON, ISTATE, NBYTE, JSTMT, 8, 22)
```

c.
```
    N=1
    CALL GURSET (-300, 200, 102B, ISTATE, NBYTE, MBYTE)
    CALL GUAN(9HSTATEMENT,N,ISTATE,NBYTE,MBYTE)
    CALL GIDISP (NCON, ISTATE, NBYTE, JSTMT, 7, 22)
```

8. The following sets the starting point of the figure initialized by GUSEGI at absolute coordinates:

```
CALL GURSET (-200, -200, 102B, IBUF, NBYTE, MBYTE)
CALL GUSEGI (-200, -200, -0, IBUF, NBYTE, MBYTE)
```

a. (0, 0)

b. (-400, -400)

c. (-200, -200)

9. The correct code to display a solid line triangle with a vertex at absolute (200, 200) is:

a.
```
    CALL GURSET (200, 200, 103B, IBUF, NBYTE, MBYTE)
    CALL GUSEGI (0, 0, 7777B, IBUF, NBYTE, MBYTE)
    CALL GUSEG (100, -200, 1)
```

```
     CALL GUSEG (-100, -200, 1)
     CALL GUSEG(0, 0, 0)
     CALL GIDISP (NCON, IBUF, NBYTE, ITNGL, 1)
```

b.
```
     CALL GURSET (200, 200, 103B, IBUF, NBYTE, MBYTE)
     CALL GUSEGI(200, 200, 0, IBUF, NBYTE, MBYTE)
     CALL GUSEG(100, 0, 1)
     CALL GUSEG(-100, 0, 1)
     CALL GUSEG(200, 200, 1)
     CALL GIDISP (NCON, IBUF, NBYTE, ITRI, 1)
```

c.
```
     CALL GURSET(0, 0, 102B, ITRI, NBYTE, MBYTE)
     CALL GUSEGI(0, 0, -0, ITRI, NBYTE, MBYTE)
     CALL GUSEG(200, 200, 0)
     CALL GUSEG(100, 0, 1)
     CALL GUSEG(-100, 0, 1)
     CALL GUSEG(200, 200, 0)
     CALL GIDISP(NCON, ITRI, NBYTE, ITNEL, 1)
```

10. This code displays which one of the figures described below?

```
CALL GURSET (0, 0, 106B, IBUF, NBYTE, MBYTE)
CALL GUSEGS(0, 0, +200, -200, 1, 0, IBUF, NBYTE, MBYTE)
CALL GUSEG(0, -200, 0)
CALL GUSEG(200, 0, 1)
CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 1)
```

   a. A triangle with vertex at (0, 0) which blinks.

   b. A blinking X with center at (100, -100).

   c. A blinking square with sides 200 dgus long.

11. CALL GUARCG (1, 300, 300, 500, 300, 300, 500, 7777B, IBUF, NBYTE, MBYTE) –

   a. Describes a 90° arc with center (300, 500).

   b. Describes a 270° arc with center (300,300), and starting point (500, 300).

   c. Describes a solid line arc with center (300,300).

12. If you wished to create a display macro from an IBUF description buffer, you would use:

   a. CALL GUMACG (MAD1, L, IBUF, NBYTE, MBYTE)

   b. CALL GIMACE (MAD1, MAD2, ..., MADN)

   c. CALL GIMAC (NCON, IBUF, NBYTE, MAD)

13. After using CALL GUSEGS(0, 0, 600, 0, 1, 7777B, IBUF, NBYTE, MBYTE) to describe a line, which of the following should be used to store the display bytes as a macro?

   a. CALL GIMAC (NCON, IBUF, NBYTE, MAD)

   b. CALL GIMAC (NCON, IFIGURE, NBYTE, MAD)

   c. CALL GIMAC (NCON, IFIG, NBYTE, MAD)

14. Which of the following correctly completes the coding to display a circle macro ICIRCL?

```
CALL GIMAC (NCON, IBUF, NBYTE, ICIRCL)
CALL GURSET (0, 0, 103B, IBUF, NBYTE, MBYTE)
CALL        (a, b, or c below)
CALL GIDISP (NCON, IBUF, NBYTE, JCIRC, 8)
```

   a. CALL GUMACG (ICIRCL, 1, IBUF, NBYTE, MBYTE)

   b. CALL GUMACG (ICIRCL, 3, IBUF, NBYTE, MBYTE)

   c. CALL GIMAC (NCON, IBUF, NBYTE, ICIRCL)

15. To erase the item displayed in Problem 14, which of the following should be used?

   a. CALL GIMACE (JCIRC)

   b. CALL GIMACE (ICIRCL)

   c. CALL GIERAS (JCIRC)

16. CALL GIMACE (MAD1, MAD2, MAD3) is used to:

   a. Clear three macros, MAD1, MAD2, and MAD3 from the 1774's core display buffer area.

   b. Clear the screen of three macros displayed with GIDISP.

   c. Erase the display items, MAD1, MAD2, and MAD3 that are alphanumeric information.

CODING PROBLEMS

Assume the following. Masks are:

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker
MBYTE	=	310
NCON	=	1

1. Make the triangle created in Chapter 2 Coding Problem 4 a macro and display it.

2. Set up masks where:

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker mask

   assuming masks are now set to zero.

3. CALL GIDISP (NCON, IBUF, NBYTE, SQR, 2) displays a square as a single pick item, with upper right corner at (-700, 100). Provide the calls which will copy this square and display its upper right-hand corner at location (500, 0) if the item's point of origin is its upper right corner.

4. Turn the tracking cross on so the person at the console can move it to another location on the screen. Find out where the tracking cross is after a button containing IDDC=200 is picked. Wait for this button.

5. Display a circle with starting point and endpoint at the location of the moved tracking cross in Problem 4. Let 550 be the radius of the circle.

6. Use three statements to display CLEAR as a button with IDDC=2, IDWA=400, IDWB=/CLER/and C at (-1000, -1000). Do not use ENCODE.

7. Use GIBUT to get back a button pick of CLEAR from Problem 6. Use a check to determine whether GIBUT returned a pick of CLEAR or a pick of another button on the screen.

8. Turn the tracking cross on. Attach it to the display item with associative address IFENDER.

9. Turn the tracking cross on. Attach it to the macro of a wheel hub. Assume a wheel has been drawn in the following order: RIM(MAD(1)), SPOKES(MAD(2)), then HUB(MAD(3)).

10. Erase the macros in Problem 9, and the display of the wheel (WHEEL).

11. At the end of the program use GICNRL to clear the queues and release the console.

12. a. Display a circle with a radius of 100 and center at location (-700, 0).

    b. Move the circle to a position where it is centered at (400, -200).

    c. Display the abort message, MOVE COMPLETED, and terminate this job.

13. Use GILPKY to obtain the new coordinates of a tracking cross that has been moved more than 100 dgus in any direction. The old coordinates are IH and IV.

MENU EXAMPLE

GENERAL

The following Menu example is included to illustrate the use of button picks, string picks, and single picks.

Figure 4-1 is a display of a menu with some instructions. The operator is to use the lightpen to choose the APPETIZER heading after one item in the appetizer column. He chooses the ENTREE heading after one entree and the DESSERT heading after one dessert. The user may choose, one, two, three, or four of the items in the DRINK column before the heading pick. As the user makes his choices, the items that he did not choose are erased from the screen. Thus, his entire order is displayed when he has made all his choices. A pick of the NEXT ORDER button at the bottom of the screen restores the menu so another person can order a meal. The user must follow the directions carefully as the routines contain few internal checks.

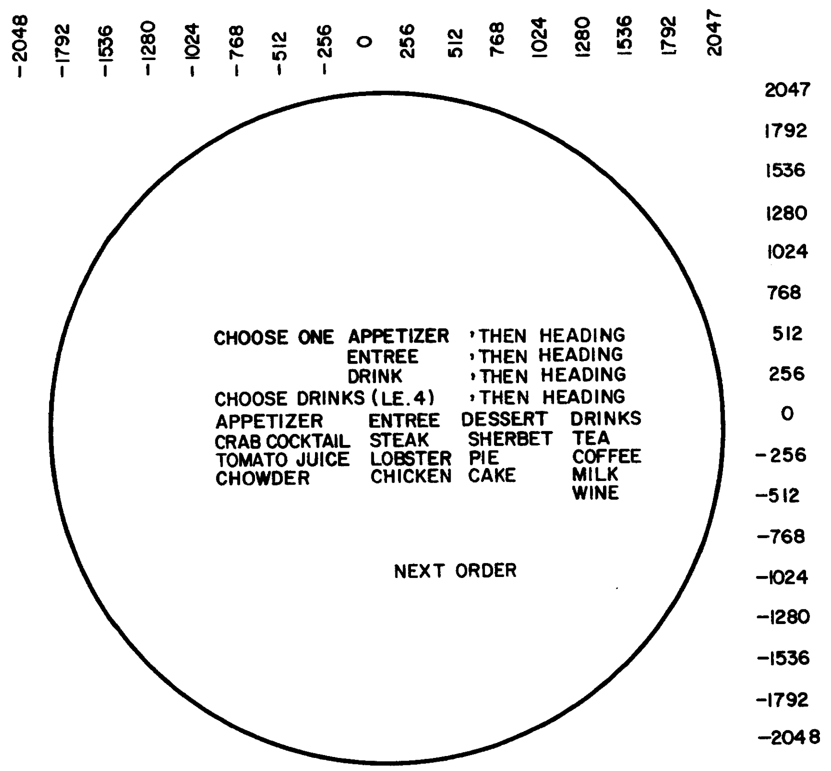


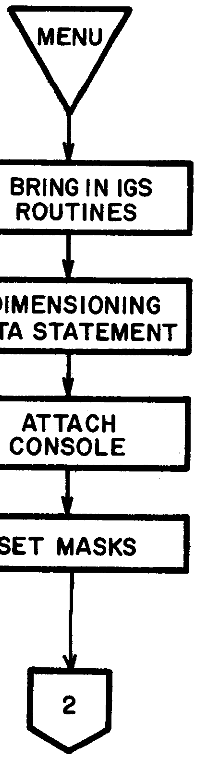
Figure 4-1. Menu

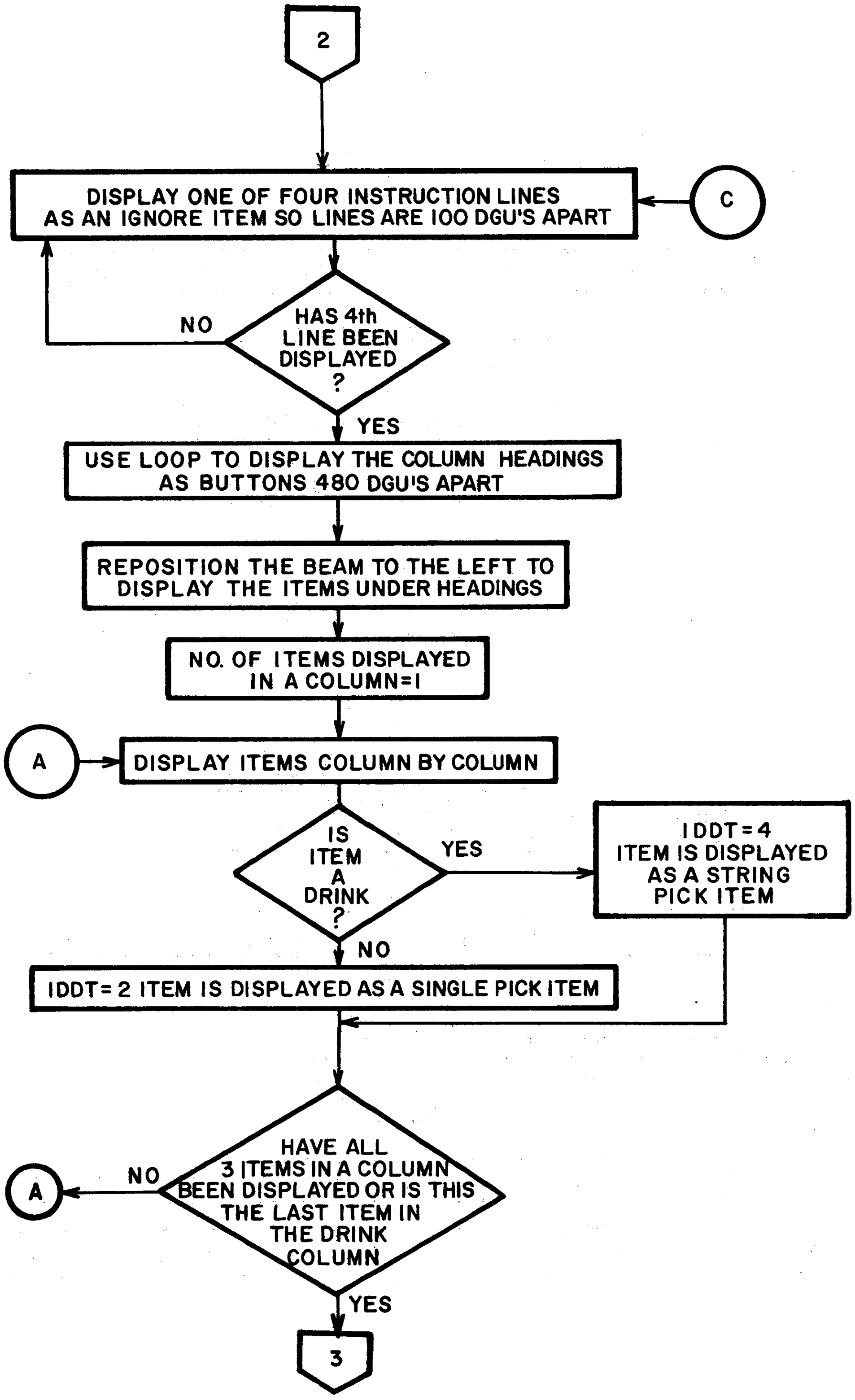
After a thorough study of the program, it is a beneficial exercise to add the necessary checks. The IDDC and IDDT of each pick returned should be checked. If there is an error, an error message should be output using GURSET, GUAN, and GIDISP. Then a jump to 191 should be made to wait for a pick of the button NEXT ORDER. The operator is then able to begin his order again. This is one of the many ways to program a path of recovery for the 274 operator.

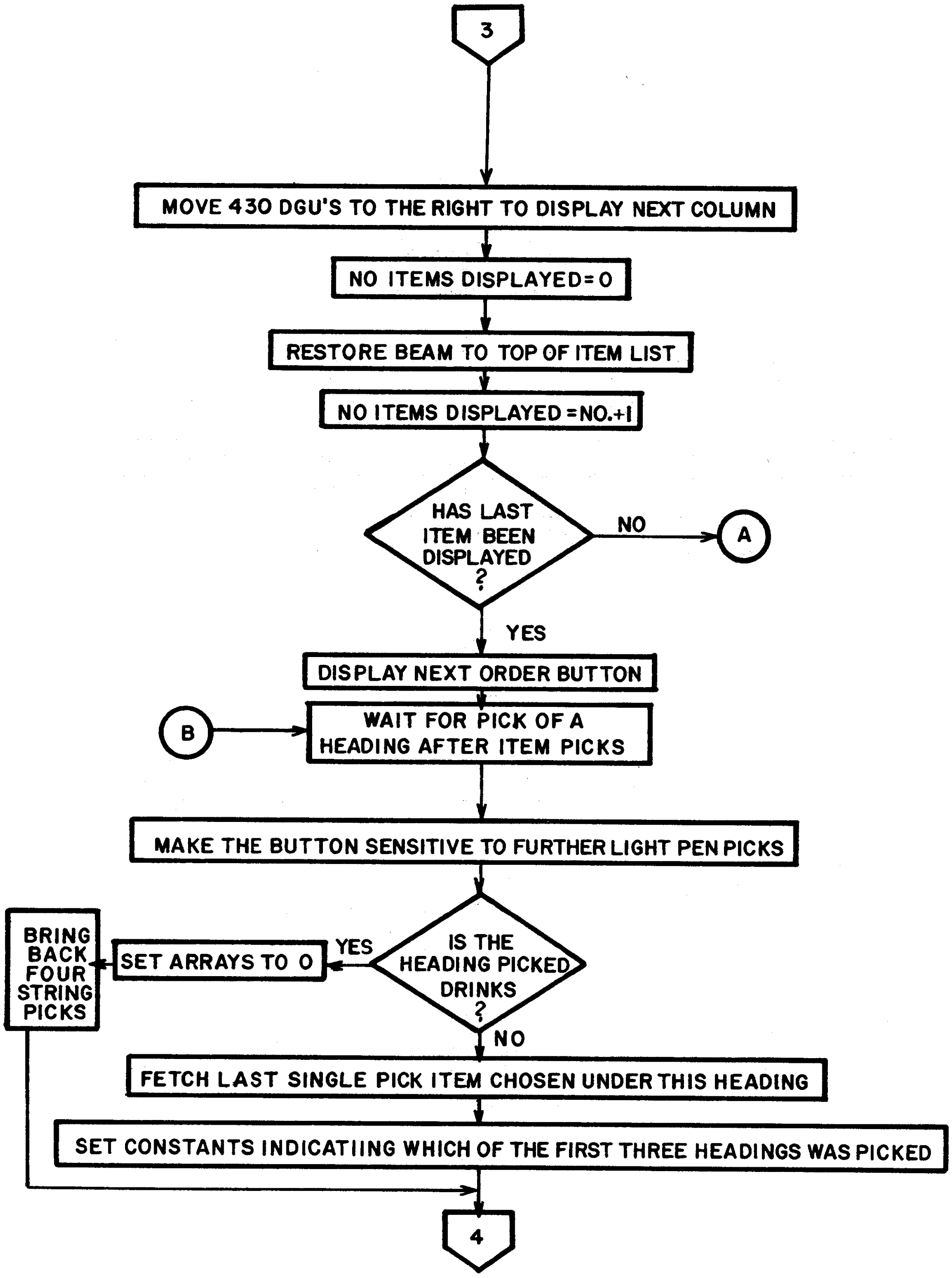
It is recommended that a button be added to voluntarily terminate the job, display a message, and release the console. A message indicating the job has been terminated can be easily displayed using GURSET, GUAN, and GIDISP.

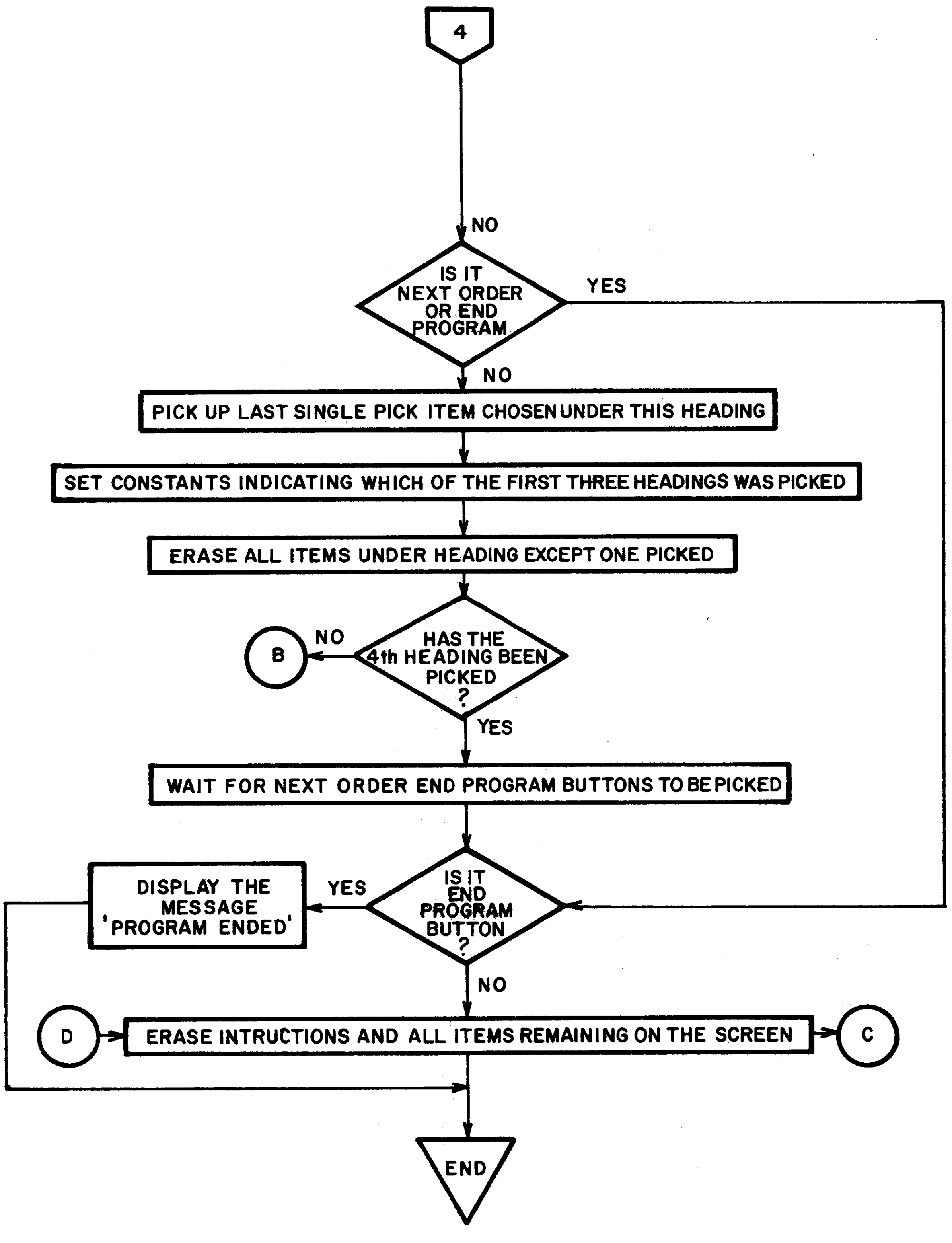
The instruction lines can be displayed as ignore items, the headings as buttons, the items under APPETIZER, ENTREE and DESSERT as single pick items, the items under DRINK as string pick items, and NEXT ORDER as a button. Assume the standard mask settings with 1 = ignore, 2 = single pick, 4 = string pick, 8 = button pick, and 16 = marker. The work area of the screen is a rectangle from (-1000, -600) to (200, 600) and (-1000, -600) to (200, -600).

MENU FLOWCHART




2
DISPLAY ONE OF FOUR INSTRUCTION LINES AS AN IGNORE ITEM SO LINES ARE 100 DGU'S APART
C
HAS 4th LINE BEEN DISPLAYED ?
NO
YES
USE LOOP TO DISPLAY THE COLUMN HEADINGS AS BUTTONS 480 DGU'S APART
REPOSITION THE BEAM TO THE LEFT TO DISPLAY THE ITEMS UNDER HEADINGS
NO. OF ITEMS DISPLAYED IN A COLUMN=1
A
DISPLAY ITEMS COLUMN BY COLUMN
IS ITEM A DRINK ?
YES
IDDT=4 ITEM IS DISPLAYED AS A STRING PICK ITEM
NO
IDDT=2 ITEM IS DISPLAYED AS A SINGLE PICK ITEM
HAVE ALL 3 ITEMS IN A COLUMN BEEN DISPLAYED OR IS THIS THE LAST ITEM IN THE DRINK COLUMN
NO
A
YES
3


3
MOVE 430 DGU'S TO THE RIGHT TO DISPLAY NEXT COLUMN
NO ITEMS DISPLAYED=0
RESTORE BEAM TO TOP OF ITEM LIST
NO ITEMS DISPLAYED =NO.+1
HAS LAST ITEM BEEN DISPLAYED ?
NO
A
YES
DISPLAY NEXT ORDER BUTTON
B
WAIT FOR PICK OF A HEADING AFTER ITEM PICKS
MAKE THE BUTTON SENSITIVE TO FURTHER LIGHT PEN PICKS
IS THE HEADING PICKED DRINKS ?
YES
SET ARRAYS TO 0
BRING BACK FOUR STRING PICKS
NO
FETCH LAST SINGLE PICK ITEM CHOSEN UNDER THIS HEADING
SET CONSTANTS INDICATIING WHICH OF THE FIRST THREE HEADINGS WAS PICKED
4


4
NO
IS IT NEXT ORDER OR END PROGRAM
YES
NO
PICK UP LAST SINGLE PICK ITEM CHOSEN UNDER THIS HEADING
SET CONSTANTS INDICATING WHICH OF THE FIRST THREE HEADINGS WAS PICKED
ERASE ALL ITEMS UNDER HEADING EXCEPT ONE PICKED
HAS THE 4th HEADING BEEN PICKED ?
NO
B
YES
WAIT FOR NEXT ORDER END PROGRAM BUTTONS TO BE PICKED
IS IT END PROGRAM BUTTON ?
YES
DISPLAY THE MESSAGE 'PROGRAM ENDED'
NO
D
ERASE INTRUCTIONS AND ALL ITEMS REMAINING ON THE SCREEN
C
END

MENU LOOP

The CALL GURSET, CALL GUAN, and CALL GIDISP series is used to display alphanumeric information. There are 22 display items in this problem. Therefore, it would require at least 45 lines of code just to display the menu, instructions, and NEXT ORDER button. A loop is the best approach, but remember that ENCODE does not allow a variable as the format number parameter. A data statement is used to display the desired information in a reasonable number of lines of code.

```
C    DISPLAY INSTRUCTION LINES
     DATA IBCD/40HCHOOSE ONE APPETIZER ,THEN HEADING     ,
    1          40H           ENTREE    ,THEN HEADING     ,
    1          40H           DESSERT   ,THEN HEADING     ,
    1          40HCHOOSE DRINKS(.LE.4) ,THEN HEADING     ,
    120HAPPETIZER            ,20HENTREE              ,
    120HDESSERT              ,20HDRINK               ,
    120HCRAB COCKTAIL        ,20HTOMATO JUICE        ,
    120HCHOWDER              ,20HSTEAK               ,
    120HLOBSTER              ,20HCHICKEN             ,
    120HSHERBET              ,20HPIE                 ,
    120HCAKE                 ,20HTEA                 ,
    120HCOFFEE               ,20HMILK                ,
    120HWINE                 ,20HNEXT ORDER          /
500  IV=550
     DO 20 I=1,13,4
     CALL GURSET(-840,IV,102B,IBUF,NBYTE,MBYTE)
     CALL GUAN(IBCD(I),40,IBUF,NBYTE,MBYTE)
     CALL GIDISP(NCON,IBUF,NBYTE,IER(I),1,I)
20   IV=IV-100
C    DISPLAY MENU LABELS AS BUTTONS
     IH=-960
     DO 30 J=17,23,2
     CALL GURSET(IH,50,102B,IBUF,NBYTE,MBYTE)
     CALL GUAN(IBCD(J),13,IBUF,NBYTE,MBYTE)
     CALL GIDISP(NCON,IBUF,NBYTE,IER(J),8,J)
30   IH=IH+480
     IH=-960
     IV=-50
     IDDT=2
     ICNT=1
C    DISPLAY ALL ITEMS ON THE MENU AS SINGLE PICKS EXCEPT DRINKS
C    DRINKS ARE STRING PICKS
     DO 40 K=25,49,2
     IF(K.GE.43)IDDT=4
     CALL GURSET(IH,IV,102B,IBUF,NBYTE,MBYTE)
     CALL GUAN(IBCD(K),13,IBUF,NBYTE,MBYTE)
     CALL GIDISP(NCON,IBUF,NBYTE,IER(K),IDDT,K)
     IF(ICNT.LT.3.OR.K.EQ.47)GO TO 39
     IH=IH+480
     ICNT=0
     IV=50
39   ICNT=ICNT+1
40   IV=IV-100
C    DISPLAY NEXT ORDER BUTTON
     CALL GURSET(-120,-550,102B,IBUF,NBYTE,MBYTE)
     CALL GUAN(IBCD(51),10,IBUF,NBYTE,MBYTE)
     CALL GIDISP(NCON,IBUF,NBYTE,IER(51),8,51)
```

The user now begins to use the light-pen to choose the items of a meal. The program must identify and fetch the picks the user makes. As the picks are identified, the items that were not picked are erased.

```
C      CALL FOR BUTTON ID BLOCKS FOR ALL FOUR BUTTONS
       DO 109 II=1,4
       CALL GIBUT(0,NCON,IDT,IDC)
C      CHANGE THE BUTTON TO AN IGNORE ITEM
       CALL GIMOVE(-0,-0,-0,IER(IDC),1)                    See Note 1
C      WHEN BUTTON IS FOR A STRING OF DRINKS,BRANCH
       IF(IDC.EQ.23)GO TO 102
C      CALL FOR SINGLE PICKS
       CALL GIFID(NCON,IDTT,IDCC)                          See Note 2
       GO TO 107
C      CALL FOR STRING PICKS
102    CONTINUE
C      SET ARRAYS TO ZERO
       DO 103 I=1,4                                        See Note 3
       IDTX(I)=0
103    ITCX(I)=0
       DO 104 LM=1,4
       NC=1
104    CALL GIFSID(NCON,NC,IDTX(LM),ITCX(LM))              See Note 4
C      ERASE UNPICKED DRINKS
       IIJ=0
       DO 106 J=43,49,2                                    See Note 5
       DO 105 I=1,4
       IF(J.EQ.ITCX(I))IIJ=J
       IF(IIJ.GT.0)GO TO 106
       IF(I.EQ.4.AND.IIJ.EQ.0)CALL GIERAS(IER(J))          See Note 6
105    CONTINUE
106    IIJ=0
       GO TO 109
C      ERASE ALL BUT PICKED ITEMS UNDER BUTTON             See Note 7
107    IF(IDC.EQ.17)IJ=25
       IF(IDC.EQ.19)IJ=31
       IF(IDC.EQ.21)IJ=37
C      ERASE ITEMS UNDER APPETIZER OR ENTREE OR DESSERT
       DO 108 JJ=1,5,2
       IK=JJ+IJ-1
       IF(IK.EQ.IDCC)GO TO 108
       CALL GIERAS(IER(IK))
108    CONTINUE
109    CONTINUE
```

Note 1: In order to change an item displayed as a button to an item displayed as an ignore item, the IDDT parameter must be changed from 8 to 1. The CALL GIMOVE(-0, -0, -0, IER(IDC), 1) says the display items IH, IV, and ICODE parameters remain the same. IDDAD is IER(IDC) and the new IDDT=1. GIMOVE can change the location, reset sequence and/or ID block information of an item. The intensity or light-pen sensitivity of an item might be changed. The first four parameters are required. (See Appendix A for additional uses of GIMOVE.)

Note 2: CALL GIFID(NCON, IDDT, IDCC) fetches the last single pick ID block stored in the 6000 series buffer. In this case, and in most cases, it is the ID block of the last single pick item chosen by the light-pen. GIFID destroys the queued ID block.

Note 3: The string pick items that will be returned have different IDDC values, so an IDDC array is dimensioned. Some may be marked string picks with a value of 24B, while other items may be unmarked with an IDDT = 4, so you must also dimension and set to zero an array for IDDT.

Note 4: The call statement format for GIFSID is CALL GIFSID(NCON, N, IDDT, IDDC, IDWA, IDWB, IH, IV) with the first three parameters required. N is the number of string pick type ID blocks the programmer wishes to fetch. Only 16 string pick blocks at a time are queued in the buffer. If fewer than N blocks are queued, N is returned equal to the number of blocks that were actually fetched. When N is greater than one, the programmer must dimension the parameters IDDT, IDDC, IDWA, IDWB, IH, and IV. IDDT and IDDC are returned as a result of the call. The horizontal and vertical coordinates, IH and IV, are the coordinates of the light beam register when the interrupt (which caused the block to be queued) occurred. They are returned as a result of the call. IH and IV are in the vicinity of the initial point of the display item and they probably vary from pick to pick. See Appendix A example, DISPLAY AN ID BLOCK RETURNED FROM A CALL TO GIFSID(A-6).

Note 5: The nested DO loops compare the IDDC code of each drink with the IDDCs of the drinks chosen. When a match is found, comparisons of the next IDDCs begin. If no match is found for a particular IDDC, the item with that IDDC is erased.

Note 6: The unpicked drink is erased and GIERAS returns 0 to the IDDAD of the item, which in this case is IER(J). Later, when the entire screen is erased, the IER(J) = 0 will not be processed, but later items will be erased if IER(J) is nonzero.

Note 7: This section of code determines which button was returned. A button with an IDC of 9 is the APPETIZER button since the GIDISP, which originally displayed APPETIZER, had an IDDC = 9. Thus, a button with an IDC = 1 is the ENTREE button.

The following code fetches the NEXT ORDER button and erases the rest of the console. The program then loops back to the display section to display the menu for the next order.

```
C    ERASE REST OF CONSOLE AFTER NEXT ORDER BUTTON
C    PICK
191  CALL GIBUT(0,NCON,IDUM,IDX)
     IF(IDX.NE.22)GO TO 191
     DO 190 I=1,43,2
190  CALL GIERAS(IER(I))
     GO TO 500
```

CREATION RUN

When running an IGS program, it is necessary to make a creation run which assigns the program to a file. This is followed with as many execution runs as desired. The creation run deck is set up in the following way.

```
     MENU,P17,T10000,CM40000.
     RUN(S)
     LGO.
     AEFILE.
     7-8-9
        OVERLAY(SOURCE,0,0)
        PROGRAM CREATE(INPUT,OUTPUT)
        CALL MAIN
        END
        OVERLAY(1,0)
        PROGRAM NMENU
        DIMENSION IER(45),IDTX(4),ITCX(4),IBCD(45),IBUF(100)
        NCON=1
        NBYTE=0
        MBYTE=310
        CALL GICNJB(NCON)
        DO 26 I=1,5
        J=2**(I-1)
26      CALL GIMASK(NCON,0,J,J)
          •
          •
          •
        PROGRAM MENU
          •
          •
          •
        END
     7/8/9
     OBJECT
     SOURCE
     6/7/8/9
```

EXECUTION RUN

The execution deck for the Menu example is the following:

```
MENU,P17,T10000,CM40000.
RUN(S)
COMMON,OBJECT.
LGO.
RELEASE,OBJECT.
EXIT.
RELEASE OBJECT
7/8/9
   OVERLAY(SOURCE,0,0)
   PROGRAM CREATE(INPUT,OUTPUT)
   CALL MAIN
   END
7/8/9
OBJECT
6/7/8/9
```

The task creation and execution runs are set for a program which does not read data cards. If data cards are used with the program, the PROGRAM CREATE card in both decks must be replaced with a PROGRAM CREATE(INPUT, OUTPUT) card.

The name of the graphics common file can be changed from one job to another, but it must be identical in the creation run and in the execution run. The name of the zero-zero overlay can also change from job to job.

MENU EXAMPLE LISTING[†]

The listing of the complete Menu example with its creation deck setup and execution deck setup follows. (See page 5-10 for information regarding SCOPE 3.3 update information.)

```
MENU,P17,T10000,CM40000.
RUN(S)
LGO.
AEFILE.
7/8/9
      OVERLAY(SOURCE,0,0)
      PROGRAM CREATE(INPUT,OUTPUT)
      CALL MAIN
      END
      OVERLAY(1,0)
      PROGRAM NMENU
      DIMENSION IER(53),IDTX(4),ITCX(4),IBCD(53),IBUF(64)
      NCON=1
      NBYTE=0
      MBYTE=310
```

[†] Appendix A includes a different approach to the Menu problem.

```
      CALL GICNJB(NCON)
      DO 26 I=1,5
      J=2**(I-1)
26    CALL GIMASK(NCON,0,J,J)
C     DISPLAY INSTRUCTION LINES
     1DATA IBCD/40HCHOOSE ONE APPETIZER ,THEN HEADING
     1      ,    40H           ENTREE    ,THEN HEADING       ,
     1           40H           DESSERT   ,THEN HEADING       ,
     1           40HCHOOSE DRINKS(.LE.4) ,THEN HEADING       ,
     120HAPPETIZER           ,20HENTREE                            ,
     120HDESSERT             ,20HDRINK                 ,
     120HCRAB COCKTAIL       ,20HTOMATO JUICE          ,
     120HCHOWDER             ,20HSTEAK                 ,
     120HLOBSTER             ,20HCHICKEN               ,
     120HSHERBET             ,20HPIE                   ,
     120HCAKE                ,20HTEA                   ,
     120HCOFFEE              ,20HMILK                  ,
     120HWINE                ,20HNEXT ORDER            /
500   IV=550
      DO 20 I=1,13,4
      CALL GURSET(-840,IV,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD(I),40,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(I),1,I)
      NBYTE=0
20    IV=IV-100
C     DISPLAY MENU LABELS AS BUTTONS
      IH=-960
      DO 30 J=17,23,2
      CALL GURSET(IH,50,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD(J),13,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(J),8,J)
      NBYTE=0
30    IH=IH+480
      IH=-960
      IV=-50
      IDDT=2
      ICNT=1
C     DISPLAY ALL ITEMS ON THE MENU AS SINGLE PICKS EXCEPT DRINKS
C     DRINKS ARE STRING PICKS
      DO 40 K=25,49,2
      IF(K.GE.43)IDDT=4
      CALL GURSET(IH,IV,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD(K),13,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(K),IDDT,K)
      NBYTE=0
      IF(ICNT.LT.3.OR.K.EQ.47)GO TO 39
      IH=IH+480
      ICNT=0
      IV=50
39    ICNT=ICNT+1
40    IV=IV-100
C     DISPLAY NEXT ORDER BUTTON
      CALL GURSET(-120,-550,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD(51),10,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(51),8,22)
      NBYTE=0
C     CALL FOR BUTTON ID BLOCKS FOR ALL FOUR BUTTONS
      DO 109 II=1,4
      CALL GIBUT(0,NCON,IDT,IDC)
C     CHANGE THE BUTTON TO AN IGNORE ITEM
      CALL GIMOVE(-0,-0,-0,IER(IDC),1)
C     WHEN BUTTON IS FOR A STRING OF DRINKS,BRANCH
      IF(IDC.EQ.23)GO TO 102
C     CALL FOR SINGLE PICKS
      CALL GIFID(NCON,IDTT,IDCC)
      GO TO 107
C     CALL FOR STRING PICKS
102   CONTINUE
C     SET ARRAYS TO ZERO
      DO 103 I=1,4
      IDTX(I)=0
103   ITCX(I)=0
      DO 104 LM=1,4
      NC=1
104   CALL GIFSID(NCON,NC,IDTX(LM),ITCX(LM))
C     ERASE UNPICKED DRINKS
      IIJ=0
      DO 106 J=43,49,2
      DO 105 I=1,4
      IF(J.EQ.ITCX(I))IIJ=J
      IF(IIJ.GT.0)GO TO 106
      IF(I.EQ.4.AND.IIJ.EQ.0)CALL GIERAS(IER(J))
105   CONTINUE
106   IIJ=0
      GO TO 109
C     ERASE ALL BUT PICKED ITEMS UNDER BUTTON
107   IF(IDC.EQ.17)IJ=25
      IF(IDC.EQ.19)IJ=31
      IF(IDC.EQ.21)IJ=37
C     ERASE ITEMS UNDER APPETIZER OR ENTREE OR DESSERT
      DO 108 JJ=1,5,2
      IK=JJ+IJ-1
      IF(IK.EQ.IDCC)GO TO 108
      CALL GIERAS(IER(IK))
108   CONTINUE
109   CONTINUE
C     ERASE REST OF CONSOLE AFTER NEXT ORDER BUTTON PICK
191   CALL GIBUT(0,NCON,IDUM,IDX)
      DO 189 J=1,13,4
189   CALL GIERAS(IER(J))
      IF(IDX.NE.22)GO TO 191
      DO 190 I=17,51,2
190   CALL GIERAS(IER(I))
      GO TO 500
      END
7/8/9
OBJECT
SOURCE
```

```
6/7/8/9
MENU,P17,T10000,CM40000.
RUN(S)
COMMON,OBJECT.
LGO.
RELEASE,OBJECT.
EXIT.
RELEASE,OBJECT.
7/8/9
      OVERLAY(SOURCE,0,0)
      PROGRAM CREATE(INPUT,OUTPUT)
      CALL MAIN
      END
7/8/9
OBJECT
6/7/8/9
```

REVIEW QUESTIONS

1. If the programmer wishes to fetch a button pick ID block, he will use:

   a. CALL GIBUT (0, NCON, IDDT, IDDC, IDWA, IDWB, IH, IV)

   b. CALL GIFSID (NCON, 1, IDDT, IDDC, IDWA, IDWB, IH, IV)

   c. CALL GIFID (NCON, IDDT, IDDC, IDWA, IDWB, IH, IV)

2. Which of the following will give faulty information?

   a. CALL GIBUT (1, NCON, IDDT, IDDC)

   b. CALL GIFSID (NCON, 3, IDT)

   c. CALL GIFID (NCON, IDT, IDC, IDA, IDB, 100, 300)

3. YES and NO are displayed on the console. The programmer uses a light-pen to pick one of them. Which of the following correctly fetches a YES button or a NO button?

   a.
```
CALL GIDISP (NCON, IYES, NBYTE, MYES, 8)
 ●
 ●
 ●
CALL GIDISP (NCON, INO, NBYTE, MNO, 8)
CALL GIBUT (0, NCON, 8, IDDC)
IF (IDDC. EQ. 22) GO TO 20
IF (IDDC. EQ. 33) GO TO 30
 ●
 ●
 ●
```

   b.
```
CALL GIDISP (NCON, IYES, NBYTE, MYES, 2)
CALL GIDISP (NCON, INO, NBYTE, MNO, 2)
 ●
 ●
 ●
CALL GIBUT (0, NCON, ITYP)
IF (ITYP. EQ. 8) GO TO 20
GO TO 20
 ●
 ●
 ●
```

   c.
```
CALL GIDISP (NCON, IYES, NBYTE, MYES, 8, 22)
 ●
 ●
 ●
CALL GIDISP (NCON, INO, NBYTE, MNO, 8, 33)
CALL GIBUT (0, NCON, IDDT, IDDC)
IF (IDDC. EQ. 22) GO TO 20
```

```
IF(IDDC. EQ. 33) GO TO 30
●
●
●
```

4. A CALL GIBUT –

   a. Precedes a CALL GIFSID when GIFSID is to process a button pick.

   b. Starts an ID block and a CALL GIFID processes the ID parameter from the last string pick.

   c. Requires that the first four parameters of its parameter list be specified.

5. What is a light register?

   a. A bright block of light that appears on the console screen at a given location.

   b. The button on the head of the light-pen that is present for a light-pen pick.

   c. An area on the screen defined by underlines and used to input and fetch alphanumeric information.

6. CALL GIERAS(IDAD1, IDAD2, IDAD3) –

   a. Erases the display items with IDDADs equal to IDAD1, IDAD2, IDAD3, and returns -0 in the IDDAD of each display item that has been erased.

   b. Erases three display items when IDAD1 = IDAD1, IDAD2 = -0, and IDAD3 = 0.

   c. Erases two items when IDAD1 = IDAD1, IDAD2 = IDAD2, and IDAD3 = -0.

7. GICNJB is used:

   a. At the end of a program to erase a single display item.

   b. To attach a console before displaying items.

   c. To attach a console after ID blocks are filled.

8. GICLR is useful for:

   a. Preventing access to previously queued ID blocks

   b. Erasing a single display item on the screen.

   c. Erasing macros.

9. GICNRL can be used:

   a. In the middle of a program to abort the job.

   b. To abort a job if the requested console number is not valid.

   c. To release the graphics console from control of the calling job.

10. The program card in the zero-zero overlay must indicate the input/output files (i.e., PROGRAM CREATE(INPUT, OUTPUT)):

    a. Whenever a READ or a WRITE are used in the IGS program.

    b. Whenever the 274 Console is used to display items.

    c. Whenever interaction occurs between the program and user.

11. What is needed to run IGS jobs?

    a. SC1700, 6000 series computer, and 250 Console.

    b. A 6000 series computer, 24K 1700, and 274 Console.

    c. 3200, 1700 series computer, and 274 Console.

TRUE OR FALSE

12.____The zero-level overlays of all runs of a graphics job must be identical.

13.____When an IGS run is input at the remote 1700 series computer site, the information goes directly to the 1700 series software package, bypassing the buffer translator and the 6000 series computer.

14.____EXPORT performs data communication between the 6000 series computer and the 1700 series computer.

15.____The IGS task level overlay cards are conventionally like RUN FORTRAN task level overlay cards, which specify only the primary and secondary overlay levels in octal.

16.____Standard 6000 series computer binary cards can be input to IGS with the proper header cards.

17.____The program card in the (1, 0) overlay may be used to list file requirements such as INPUT, OUTPUT, or TAPE 6.

CODING PROBLEMS

Assume masks are set:

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker
NCON	=	1
MBYTE	=	310

1. The masks are set as in Chapter 3 Problem 2, except the marker mask does not mark an ignore item. Format a GIDISP call that will cause a single pick item to blink when picked.

2. Suppose a programmer is beginning a program and wishes to attach a graphics console so he can display some items. Attach a console. Disconnect the console.

```
PROGRAM DISPLY
NCON = 1
         ?
         ?
         ?
```

3. Use GIFID to retrieve the approximate coordinates of the last queued single pick item in a program. Use standard parameter names.

4. Suppose there is a button pick item and three string pick items displayed. Fetch the ID parameters from the last string pick the user chose.

5. Use GIBUT to return a button. Check to see if a single pick is available. Check to see how many string picks are available.

SCISSORING

This chapter describes an example of scissoring, the input of information through font picks, and overlay structure. The problem is outlined in four steps.

PROBLEM DESCRIPTION

Step 1. Display a 10 by 10-inch square (Figure 5-1) with its center coordinates at (0,0). Display an arc with endpoints of (1600, 0) and (0, -1600) and center coordinates of (1080, -1080), of which only the scissored portion is displayed in a counterclockwise path of illumination. A line with endpoints of (1200, -24) and (-800, 16) is described but, once again, only the scissored portion is illuminated.

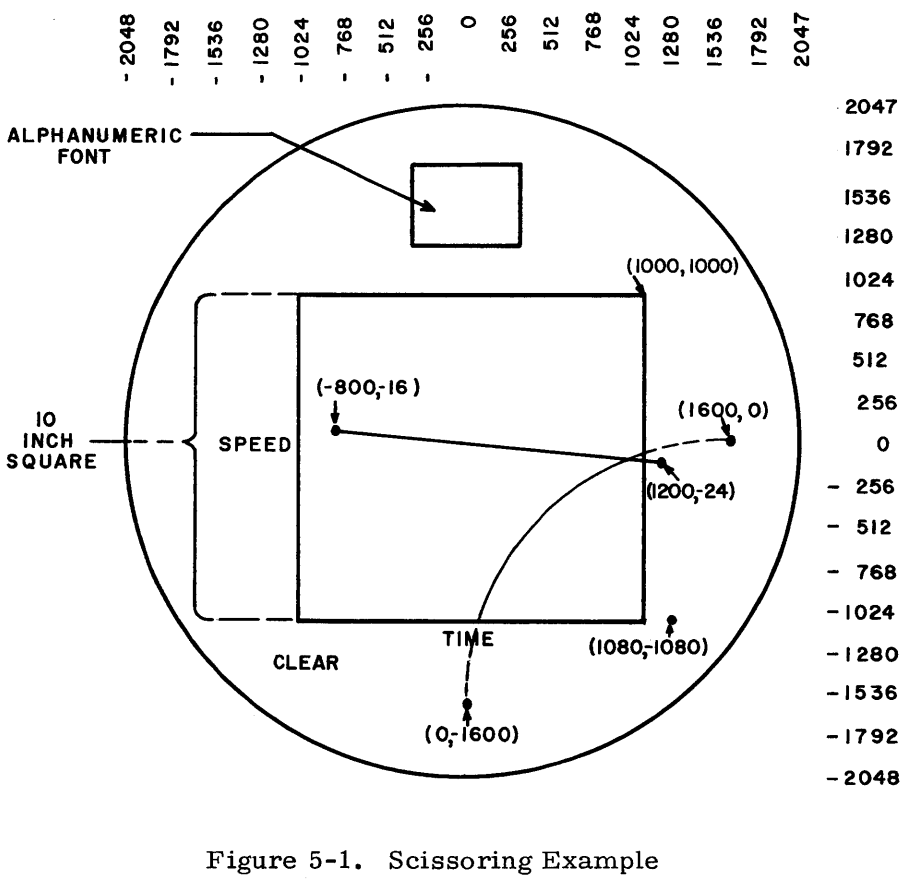


Figure 5-1. Scissoring Example

Step 2. Display an alphanumeric font at (0, 1500) with $ as the end-of-message character. Use OVERLAY (1, 1) to complete Step 2.

Step 3. Use a light register to input up to six characters for the x-axis label. Display the label with the leftmost character at (-48, -1200). Accept up to six characters for the y-axis label. Display the label with leftmost character at (-1150, 0).

Step 4. Erase the font and release the console.

The programmer should insert code to display a message when the CLEAR button is picked. The first cards of the program are

```
        OVERLAY(ABC,0,0)
        PROGRAM CREATE
        COMMON IA,IB,IC,NCON,MBYTE
        CALL MAIN
        END
        OVERLAY(1,0)
        PROGRAM SCISR
        COMMON IA,IB,IC,NCON,MBYTE
        DIMENSION IBUF(100)
        MBYTE=310
        NBYTE=0
        NCON=1
C       ATTACH CONSOLE
        CALL GICNJB(NCON)
C       SET MASKS
        DO 101 I=1,5
        N=2**(I-1)
101     CALL GIMASK(NCON,0,N,N)
C       DISPLAY CLEAR BUTTON
        CALL GURSET(-1200,-1200,102B,IBUF,NBYTE,MBYTE)
        CALL GUAN(5HCLEAR,5,IBUF,NBYTE,MBYTE)
        CALL GIDISP(NCON,IBUF,ICLR,30B,44)
```

DISPLAY SQUARE

The code for displaying the 10 by 10-inch square is similar to the code for displaying the perimeter of the winding road sign. If the center of the square is at (0, 0), the upper right-hand corner will be located at (1000, 1000) on the screen since 200 dgus/inch times 5 is 1000 dgus.

```
 CALL GURSET(1000,-1000,102B,IBUF,NBYTE,MBYTE)
 CALL GUSEGS(1000,-1000,-1000,-1000,1,7777B,IBUF,NBYTE,
1MBYTE)
 CALL GUSEG(-1000,1000,1)
 CALL GUSEG(1000,1000,1)
 CALL GUSEG(1000,-1000,1)
 CALL GIDISP(NCON,IBUF,NBYTE,IFRAM,1)
```

SCISSOR ARC

The problem is to scissor an arc and display only the portion that falls within the 10 by 10-inch square or frame. The IGS routine GUARC is an arc-scissoring routine. The call statement format is

```
      CALL GUARC(IHCEN,IVCEN,IHCOR,IVCOR,HC,VC,H1,V1,
     1H2,V2,KSHOW,IHC,IVC,IH1,IV1,IH2,IV2)
```

The first two parameters, IHCEN and IVCEN, are the coordinates of the center of the frame. IHCOR and IVCOR are the coordinates of the upper right-hand corner of the frame.

HC and VC are the coordinates of the center of the circular arc to be scissored; HC and VC are floating point numbers. H1, V1 and H2, V2 are real coordinates of the right and left ends, respectively, of the arc to be scissored.

KSHOW is a flag. When KSHOW=0 the given arc is completely outside of the frame. When KSHOW=1 through 5, it indicates the number of arc segments within the frame.

IH1, IV1, IH2, and IV2 are returned as a result of the call. These parameters form the end-points of those portions of the arc within the frame. Each of these parameters is the first word in any array KSHOW words long.

The code to scissor and display the arc in the example is

```
      CALL GUARC(0,0,1000,1000,1080,-1080.,1600.,0.,0.,-1600.,
     1KSHOW,IHC,IVC,IH1,IV1,IH2,IV2)
      IF(KSHOW.EQ.0)GO TO 1500
      CALL GURSET(IH1,IV1,102B,IBUF,NBYTE,MBYTE)
      CALL GUARCG(KSHOW,IHC,IVC,IH1,IV1,IH2,IV2,7777B,
     1IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IARC,1)
 1500 CONTINUE
```

SCISSOR LINE SEGMENT

Similarly, a line segment can be scissored and displayed. The IGS routine to scissor a line is GULINE. The call statement format is

```
      CALL GULINE(IHCEN,IVCEN,IHCOR,IVCOR,H1,V1,H2,V2,
     1KSHOW,IH1,IV1,IH2,IV2)
```

The first four parameters are the same as the first four parameters described under GUARC. H1, V1 and H2, V2 are floating point number coordinates of the left and right ends, respectively, of the line to be scissored.

If KSHOW=0, the line segment is completely outside the frame. If KSHOW=1, the line segment is completely within the frame. If KSHOW=2, the line segment is partially within the frame and has been scissored.

IH1, IV1 and IH2, IV2 are the display grid coordinates of the left and right endpoints, respectively, of the portion of the line within the frame. The coordinates are returned as a result of the call.

The code to scissor and display the line is:

```
       CALL GULINE(0,0,1000,1000,-800.,16.,1200.,-24.,KSHOW
      1JH1,JV1,JH2,JV2)
       IF(KSHOW.NE.2)GO TO 1600
       CALL GURSET(JH1,JV1,102B,IBUF,NBYTE,MBYTE)
       CALL GUSEGS(JH1,JV1,JH2,JV2,1,7777B,IBUF,NBYTE,
      1MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,LIN,1)
 1600  CONTINUE
```

ALPHANUMERIC FONT

In Step 2 an alphanumeric font (Figure 5-2) is displayed by a program in OVERLAY (1, 1). The name of the program in OVERLAY(1, 1) is FONT. The program from OVERLAY (1, 0) is called with CALL AETSKC(4LFONT) or CALL AETSKC(4HFONT). See Appendix A for the example Creation of a Light Button Under Application Executive. When the button is picked, the application executive brings task DISP into core for execution.

The vertical dimensions are IV+172 to IV-172 and the horizontal dimensions are IH-332 to IH+332. There are two IDDADs associated with this font. (This is important when erasing the font; both sections identified by IDDADs must be erased.)

To display this font (Figure 5-2), use IGS routine GFONTA. The call statement format is:

```
CALL GFONTA(NCON,IH,IV,IDDA,IDDN)
```

IH and IV are coordinates of the approximate center of the font. IDDA and IDDN are associative addresses of the display font sections and are returned as a result of the call.

END-OF-MESSAGE CHARACTER

In order to determine the end of a character string which the user is inputting with light-pen picks of characters of the font, he may establish one character as an end-of-message character. Let $ be the EOM character for the scissoring example. CALL GIEOM(NCON, IBCD, IDDT, IDDC, IDWA, IDWB) is the general call statement format. The statement in the scissoring example is CALL GIEOM(NCON, IH$, 8, 11).

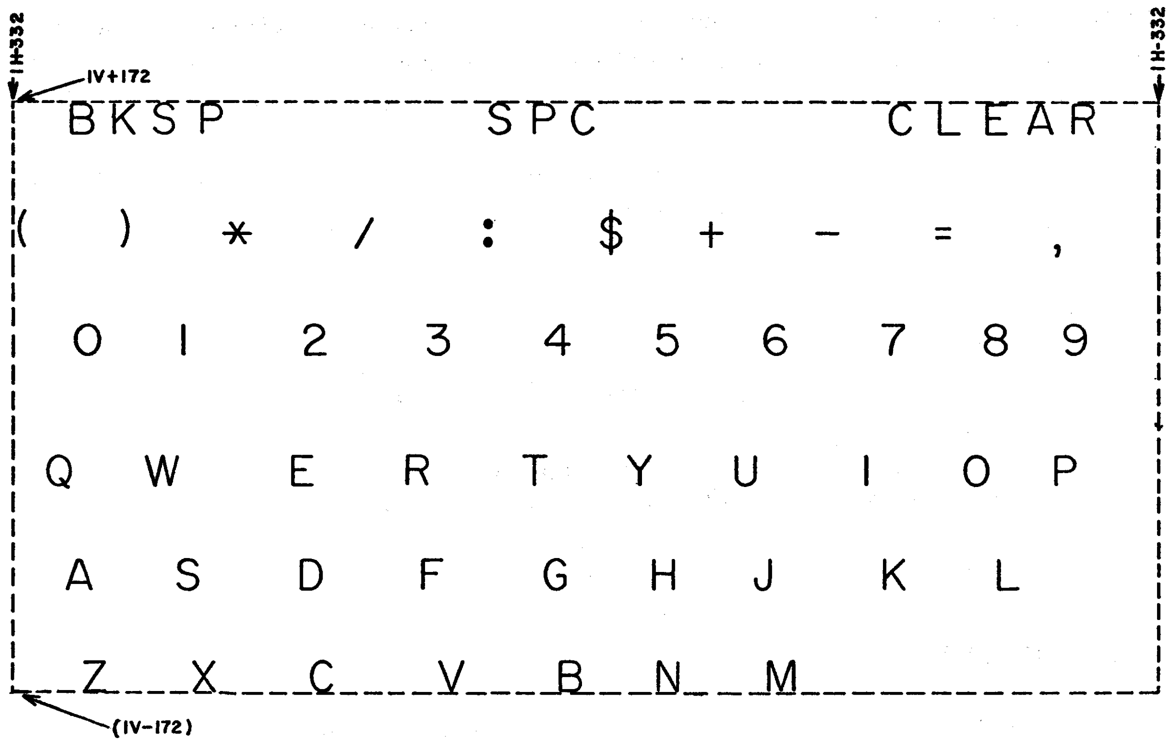


Figure 5-2. Alphanumeric Font

The code for OVERLAY(1,1) is

```
      OVERLAY(1,1)
      PROGRAM FONT
      COMMON IA,IB,NCON,MBYTE
C     SET BUTTON MASK
      CALL GIMASK(NCON,0,8,8)
C     DISPLAY FONT
      CALL GFONTA(NCON,0,1500,IA,IB)
      CALL GIEOM(NCON,1R$,8,11)
C     RETURN TO OVERLAY(1,0)
      CALL AERTRN
      END
```

CALL AERTRN returns to the statement following the CALL AETSKC card.

LIGHT REGISTER INPUT

Step 3 is coded in OVERLAY(1,0). A light register is a specific area on the screen for operator input of alphanumeric information. A CALL GIANS(NCON, NC, IH, IV) creates a light register which will accept up to NC characters. The end-of-message character is counted as one of the characters. The user sees a dashed line at the location of the light register, and the left-hand end of the dashed line has coordinates IH, IV.

Information may be retrieved from the light register. Suppose the user is picking the letters SPEED$ with the light-pen. Check the input again and again for a button with IDDC=11, $, with a call GIBUT. When $ is chosen, call GIANE to retrieve the information from the light register. CALL GIANE(NCON, NC, IBUF) brings back NC characters and stores them in IBUF. In this case, the user will input axis names of variable lengths so that the NC returned is the actual number of characters chosen. The $ is usually not displayed as part of the label, so let NC=NC-1 for the number of characters to be displayed.

The code for Step 3 is

```
              K=0
      1000    NC=6
              CALL GIANS(NCON,6,1400,1200)
      1100    CALL GIBUT(0,NCON,IDDT,IDDC)
              IF(IDDC.NE.11)GO TO 1100
              CALL GIANE(NCON,NC,IBCD)
      C       TEST FOR AT LEAST ONE CHARACTER
              IF(NC.EQ.1)GO TO 1000
              NC=NC-1
              IF(K.NE.0)GO TO 1200
      C       DISPLAY AXIS LABELS ****X THEN Y
              CALL GURSET(-48,-1200,102B,IBUF,NBYTE,MBYTE)
              CALL GUAN(IBCD,NC,IBUF,NBYTE,MBYTE)
              CALL GIDISP(NCON,IBUF,NBYTE,IXAX,1)
              K=1
              GO TO 1000
      1200    CALL GURSET(-1150,0,102B,IBUF,NBYTE,MBYTE)
              CALL GUAN(IBCD,NC,IBUF,NBYTE,MBYTE)
              CALL GIDISP(NCON,IBUF,NBYTE,IYAX,1)
```

Step 4 includes erasing the font. Remembering that the alphanumeric font has two IDDADs, the user enters CALL GIERAS(IA, IB).

NUMERIC FONT

There is also a numeric font (Figure 5-3). It has only one IDDAD and its call statement format is

```
        CALL GFONTN(NCON, IH, IV, IDDAD)
```

The vertical dimensions are IV+334 and IV-244 and its horizontal dimensions are IH+268 to IH-244. The IH and IV coordinates specified are the coordinates of the center of the circle.

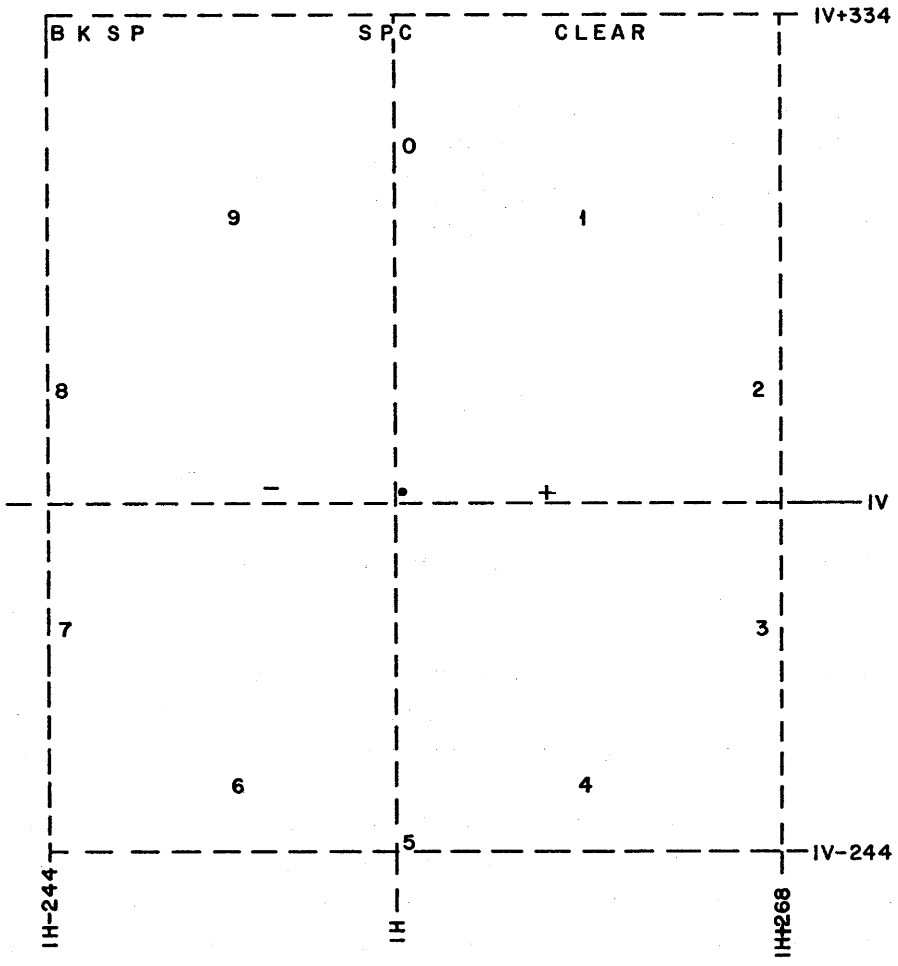


Figure 5-3. Numeric Font

ESTIMATING NBYTE

When displaying such items as the instruction for this example, there is no need for each line to be displayed with a GURSET, GUAN, GIDISP sequence of calls. All 80 characters of the instructions can be packed in one buffer IBUF, and displayed with a single GIDISP call. Refer to Appendix G of the IGS Reference Manual. The number of bytes packed by the following code may be calculated.

```
        CALL GURSET(-650,-1400,103B,IBUF,NBYTE,MBYTE)
        CALL GUAN (INST(1),40,IBUF,NBYTE,MBYTE)
        CALL GURSET(-650,-1500,103B,IBUF,NBYTE,MBYTE)
        CALL GUAN(INST(5),40,IBUF,NBYTE,MBYTE)
        CALL GIDISP(NCON,IBUF,NBYTE,MBYTE)
```

GURSET packs 3 bytes, GUAN packs 1 + NC+1/2 bytes and GIDISP packs 10 bytes.

$$3 + \left(1 + \frac{40+1}{2}\right) + 3 + \left(1 + \frac{40+1}{2}\right) + 10 = N$$

$$3 + \quad 22 \quad + 3 + \quad 22 \quad + 10 = 60 \text{ bytes} = \text{NBYTE}$$

If the fraction NC + 1 is an odd number, round the fraction NC + 1/2 up. MBYTE was set at 310 and NBYTE does not exceed MBYTE so only one GIDISP call is necessary.

In the Menu example the menu items were displayed with separate GIDISP calls because each item needed a different IDDAD so it could be identified and erased at the appropriate time. Could the four instruction lines be displayed as one item? If so, what would the code be?

If one of the lines of instruction in the Scissoring example had been a button pick item and the other an ignore item, two GIDISP calls would be required to set the two IDDT values. Whenever NBYTE approaches the maximum MBYTE (310) the programmer should check the number of bytes packed in the 1744. (This can be checked by using the fourth column (Table G-1) of Appendix G of the IGS Reference Manual.)

COMPLETING EXAMPLE

The routine GICNRL releases the specified graphics console from the control of the calling job. The user may also wish to display a message indicating the job is terminated and the console is released. The code for Step 4 is

```
2000    CALL GIBUT(0,NCON,IDUM,IDX)
        IF(IDX.NE.44)GO TO 2000
        CALL GIERAS(IA,IB,ICLR,IFRAM,IARC,LIN,INFO,INFOR,IXAX,IYAX)
        CALL GICNRL(NCON)
        STOP
        END
```

SCISSORING EXAMPLE

A listing of the entire Scissoring example creation run follows.

```
JSCISR,P17,T10000,CM40000.
RUN(S)
LGO.
AEFILE.
7/8/9
      OVERLAY(ABC,0,0)
      PROGRAM CREATE(INPUT,OUTPUT)
      COMMON IA,IB,NCON,MBYTE
      CALL MAIN
      END
      OVERLAY(1,0)
      PROGRAM SCISR
      COMMON IA,IB,   NCON,MBYTE
      DIMENSION IBUF(64),IH1(5),IV1(5),IH2(5),IV2(5),JH1(5),JV1(5),
     1JH2(5),JV2(5),IBCD(3),INST(8)
      DATA INST/40HPICK 5 CHARACTERS AND $ AS X AXIS LABEL ,
     1          40HTHEN 5 CHARACTERS AND $ AS Y AXIS LABEL /
      MBYTE=310
      NBYTE=0
      NCON=1
C     ATTACH CONSOLE
      CALL GICNJB(NCON)
C     SET MASKS
      DO 101 I=1,5
      N=2**(I-1)
101   CALL GIMASK(NCON,0,N,N)
C     DISPLAY CLEAR BUTTON
      CALL GURSET(-1200,-1200,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(5HCLEAR,5,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ICLR,30B,44)
      CALL GURSET(1000,-1000,102B,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(1000,-1000,-1000,-1000,1,7777B,IBUF,NBYTE,MBYTE)
      CALL GUSEG(-1000,1000,1)
      CALL GUSEG(1000,1000,1)
      CALL GUSEG(1000,-1000,1)
      CALL GIDISP(NCON,IBUF,NBYTE,IFRAM,1)
      CALL GUARC(0,0,1000,1000,1080.,-1080.,1600.,0.,0.,-1600.,KSHOW
```

†Appendix A includes a different approach to the Scissored Arc problem.

```
     1IHC,IVC,IH1,IV1,IH2,IV2)
      IF(KSHOW.EQ.0)GO TO 1500
      CALL GURSET(IH1,IV1,102B,IBUF,NBYTE,MBYTE)
      CALL GUARCG(KSHOW,IHC,IVC,IH1,IV1,IH2,IV2,7777B,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IARC,1)
 1500 CONTINUE
      CALL GULINE(0,0,1000,1000,-800.,16.,1200.,-24.,KSHOW,JH1,JV1,
     1JH2,JV2)
      IF(KSHOW.EQ.0)GO TO 1600
      CALL GURSET(JH1,JV1,102B,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(JH1,JV1,JH2,JV2,1,7777B,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,LIN,1)
 1600 CONTINUE
      CALL AETSKC(4LFONT)
C     DISPLAY INSTRUCTIONS
      CALL GURSET(-650,-1400,103B,IBUF,NBYTE,MBYTE)
      CALL GUAN(INST(1),40,IBUF,NBYTE,MBYTE)
      CALL GURSET(-650,-1500,103B,IBUF,NBYTE,MBYTE)
      CALL GUAN(INST(5),40,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,INFOR,1)
      K=0
 1000 NC=6
      CALL GIANS(NCON,6,1400,1200)
 1100 CALL GIBUT(0,NCON,IDDT,IDDC)
      IF(IDDC.NE.11)GO TO 1100
      CALL GIANE(NCON,NC,IBCD)
C     TEST FOR AT LEAST ONE CHARACTER
      IF(NC.EQ.1)GO TO 1000
      NC=NC-1
      IF(K.NE.0)GO TO 1200
C     DISPLAY AXIS LABELS ****X THEN Y
      CALL GURSET(-48,-1200,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD,NC,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IXAX,1)
      K=1
      GO TO 1000
 1200 CALL GURSET(-1150,0,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD,NC,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IYAX,1)
 2000 CALL GIBUT(0,NCON,IDUM,IDX)
      IF(IDX.NE.44)GO TO 2000
      CALL GIERAS(IA,IB,ICLR,IFRAM,IARC,LIN,INFO,INFOR,IXAX,IYAX)
      CALL GICNRL(NCON)
      END
      OVERLAY(1,1)
      PROGRAM FONT
      COMMON IA,IB,NCON,MBYTE
C     SET BUTTON MASK
      CALL GIMASK(NCON,0,8,8)
C     DISPLAY FONT
      CALL GFONTA(NCON,0,1500,IA,IB)
      CALL GIEOM(NCON,1R$,8,11)
C     RETURN TO OVERLAY(1,0)
      CALL AERTRN
      STOP
      END
```

```
7/8/9
OBJ
ABC
6/7/8/9
JSCISR,P17,T10000,CM40000.
RUN(S)
COMMON,OBJ.
LGO.
RELEASE,OBJ.
EXIT.
RELEASE,OBJ.
7/8/9
      OVERLAY(ABC,0,0)
      PROGRAM CREATE(INPUT,OUTPUT)
      COMMON IA,IB,NCON,MBYTE
      CALL MAIN
      END
7/8/9
OBJ
6/7/8/9
```

FILE CREATION

The general organization of a file creation run deck is illustrated in Figure 5-4. Note that the overlay cards specify only the primary and secondary overlay levels in octal.

FILE EXECUTION

This is a typical program execution run deck (Figure 5-5). It follows the deck that created the graphics common file called OBJECT. The current overlay file name need not be SOURCE, but the zero-zero overlay cards must be the same for the creation and execution runs. If line printer output is requested and the program card in the zero-zero overlay of the creation run is PROGRAM CREATE(INPUT, OUTPUT), the program card in the execution run must be identical.

ONE-PASS CREATION AND EXECUTION

It is possible to assemble a deck to create, run, and execute the program in one pass through the computer. Figure 5-6 shows such a run with a graphics common file named KCB and overlay file named SCR. The advantage of a one-pass run is that only one batch control point is required.

CONVERSION FROM RUN FORTRAN TO FORTRAN EXTENDED

If a graphics FTN job under SCOPE 3.3 is to be run, replace the RUN(S) card with an FTN card and the CALL MAIN card with a CALL AEXEC card wherever they appear in the job deck.

A DATA statement that once read:

```
       DATA IBCD/40H CHOOSE ONE APPETIZER ,THEN HEADING
      1          40H              ENTREE     ,THEN HEADING        /
```

should now read:

```
       DATA (IBCD(I),I=1,13,4)/40HCHOOSE ONE APPETIZER ,THEN HEADING
      1 ,
      1           40H            ENTREE     ,THEN HEADING        /
```

to fill words IBCD(1) through IBCD(8).

While the RUN compiler accepts both ICOMP = 010000000000B and ICOMP = O010000000000, the EXTENDED compiler recognizes only B as an indication of an octal number.

Under FORTRAN EXTENDED, the DIMENSION, COMMON, EQUIVALENCE, EXTERNAL, and type statements must appear before any statement function definition, DATA, NAMELIST, or executable statements in the program. DATA statements may appear with DIMENSION, COMMON, EQUIVALENCE, or type declarations under RUN as long as any arrays used in the DATA statement are dimensioned prior to the DATA statement.

Under FORTRAN EXTENDED, a DO loop may not terminate on an IF statement (i.e., "15 IF(I.GT.8) GO TO 200" may not follow "DO 15 I=1,10").

IGS VERSION 1 TO VERSION 2 CONVERSION

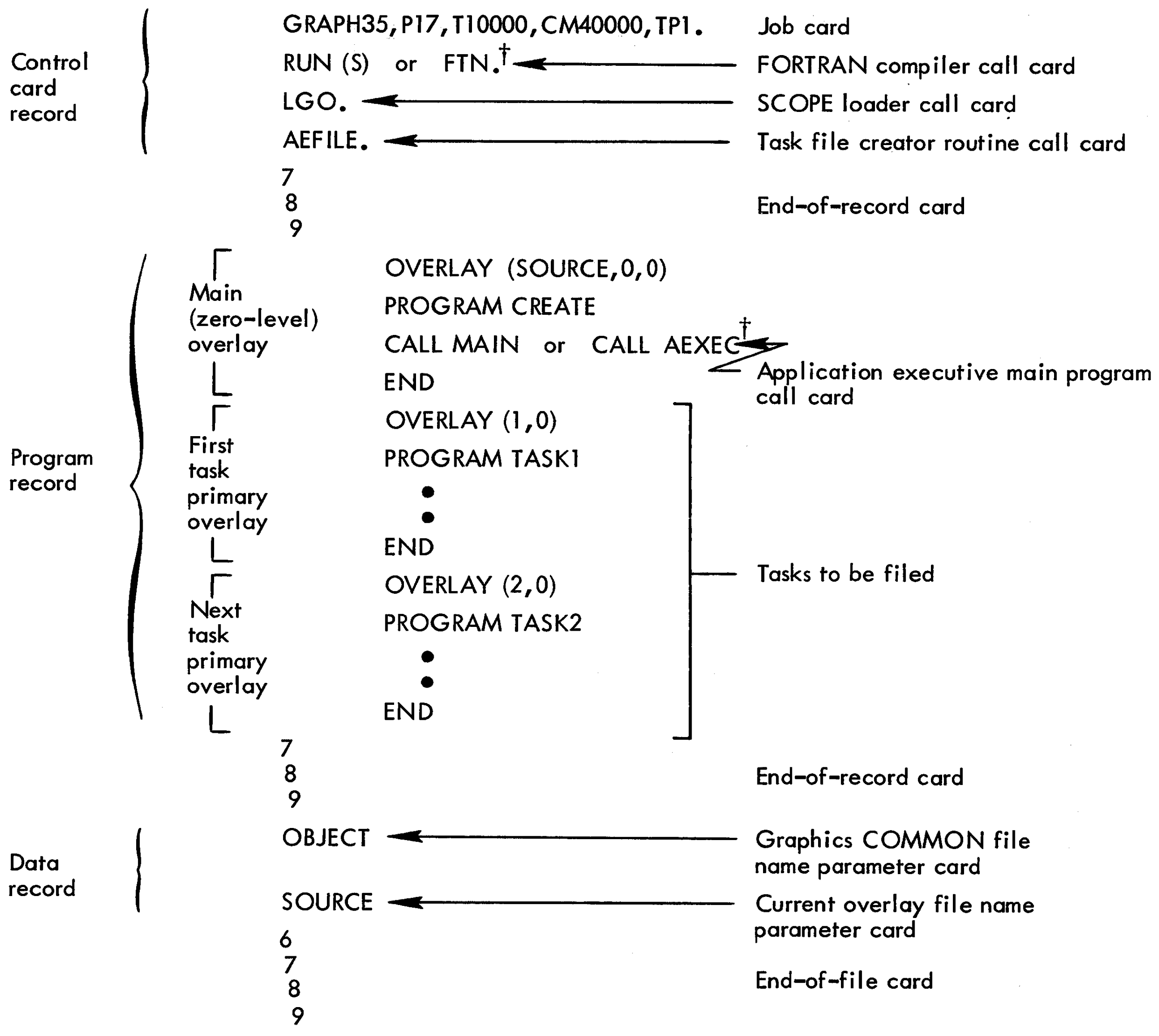
ICODE is s00tfbb where "terminate to next reset" is a hardware capability which is used to turn an item off; it is not used to remove an item from the buffer.

There is an assembly option available to convert files to local task files so that the user may subsequently convert them to permanent files. See the IGS Reference Manual for sample deck structures.

Error messages and GIABRT messages are displayed on the 274 with the leftmost character at screen coordinates (-552, 1600) rather than (0, 0).

The alphanumeric font coordinates are IH-332 to IH+332 and IV-172 to IV+172.

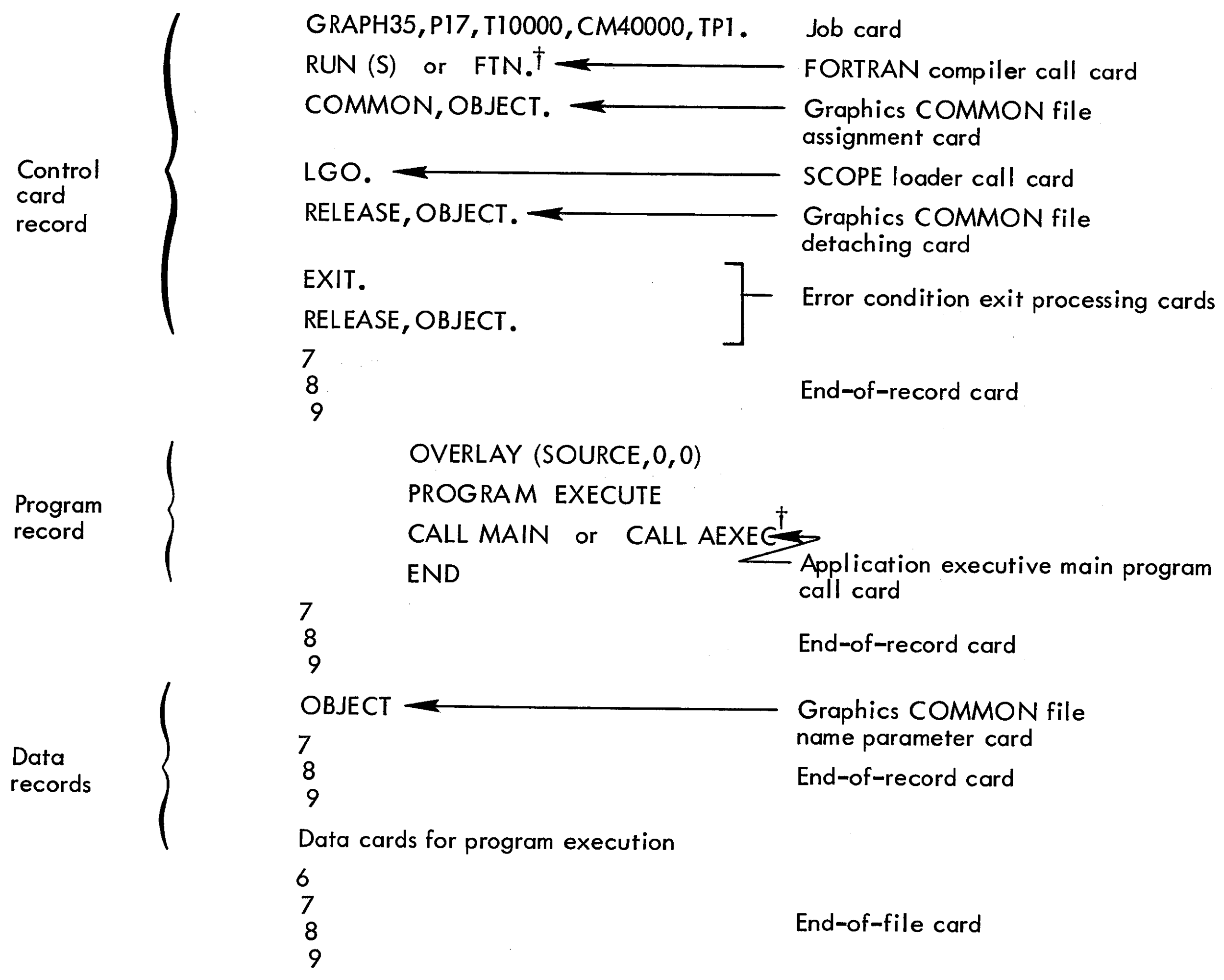
The RETURN key on the alphanumeric keyboard may be used as an end-of-message character under Version 2 of IGS.



```
Control card record  { GRAPH35,P17,T10000,CM40000,TP1.     Job card
                       RUN (S)  or  FTN.[†] <-------------- FORTRAN compiler call card
                       LGO. <------------------------------ SCOPE loader call card
                       AEFILE. <--------------------------- Task file creator routine call card
                       7
                        8                                   End-of-record card
                         9
Program record  { [ Main            OVERLAY (SOURCE,0,0)
                    (zero-level)    PROGRAM CREATE
                    overlay         CALL MAIN  or  CALL AEXEC[†] <-- Application executive main program
                  ]                 END                              call card
                  [ First           OVERLAY (1,0)        ]
                    task            PROGRAM TASK1        |
                    primary            •                 |
                    overlay            •                 |
                  ]                 END                  |
                  [                 OVERLAY (2,0)        |---------- Tasks to be filed
                    Next            PROGRAM TASK2        |
                    task               •                 |
                    primary            •                 |
                    overlay                              |
                  ]                 END                  ]
                       7
                        8                                   End-of-record card
                         9
Data record  {         OBJECT <---------------------------- Graphics COMMON file
                                                            name parameter card
                       SOURCE <---------------------------- Current overlay file name
                       6                                    parameter card
                        7
                         8                                  End-of-file card
                          9
```

Figure 5-4. File Creation Run Deck

[†] FTN is available under Version 2 and CALL AEXEC is required under Version 2.



```
Control    GRAPH35,P17,T10000,CM40000,TP1.      Job card
card       RUN (S)  or  FTN.[†]  <------------  FORTRAN compiler call card
record     COMMON,OBJECT.  <------------------  Graphics COMMON file
                                                assignment card
           LGO.  <----------------------------  SCOPE loader call card
           RELEASE,OBJECT.  <-----------------  Graphics COMMON file
                                                detaching card
           EXIT.                            ]-- Error condition exit processing cards
           RELEASE,OBJECT.                  ]
           7
           8                                    End-of-record card
           9
Program          OVERLAY (SOURCE,0,0)
record           PROGRAM EXECUTE
                 CALL MAIN  or  CALL AEXEC[†] <-
                 END                         <-- Application executive main program
                                                 call card
           7
           8                                    End-of-record card
           9
Data       OBJECT  <--------------------------  Graphics COMMON file
records    7                                    name parameter card
           8                                    End-of-record card
           9
           Data cards for program execution
           6
           7
           8                                    End-of-file card
           9
```

Figure 5-5. Execution Run Card Deck

[†] FTN is available under Version 2 and CALL AEXEC is required under Version 2.

```
          GRAPH35,P17,T10000,CM40000,TP1.     Job card
          RUN (S)  or  FTN.†  <------------   FORTRAN compiler call card
          LGO.  <------------------------     SCOPE loader call card
Control   AEFILE.  <---------------------     Task file creator routine call card
card      SOURCE.  <---------------------     Source file call card
record    RELEASE, OBJECT.
          EXIT.                          ]
          RELEASE,OBJECT.                ]--  Error condition exit processing cards
          7
          8                                   End-of-record card
          9
        ┌             OVERLAY (SOURCE,0,0)
        Main          PROGRAM CREATE (INPUT,OUTPUT)
        (zero-level)
        overlay       CALL MAIN  or  CALL AEXEC†
        └             END
        ┌             OVERLAY (1,0)           ┐
Program First         PROGRAM TASK1           |
record  task             •                    |
        primary          •                    |
        overlay       END                     |
        └                                     |-- Tasks to be filed
        ┌             OVERLAY (2,0)           |
        Next          PROGRAM TASK2           |
        task             •                    |
        primary          •                    |
        overlay       END                     ┘
        └
          7
          8                                   End-of-record card
          9
          OBJECT  <----------------------     Graphics COMMON file
                                              name parameter card
          SOURCE  <----------------------     Current overlay file name
                                              parameter card
          7
Data      8                                   End-of-record card
records   9
          OBJECT  <----------------------     Graphics COMMON file
                                              name parameter card
          7
          8                                   End-of-record card
          9
          Data cards for program execution
          6
          7
          8                                   End-of-file card
          9
```

Figure 5-6. Creation and Execution Deck

† FTN is available under Version 2 and CALL AEXEC is required under Version 2.

REVIEW QUESTIONS

Assume masks are set:

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker
NCON	=	1
MBYTE	=	310

1. The IGS routine to frame-scissor an arc is:

   a. GUARC

   b. GUARCG

   c. GIARC

2. The parameters of the arc-scissoring routine which are named IH1, IV1, IH2, IV2 are meaningless when:

   a. KSHOW=2

   b. KSHOW=1

   c. KSHOW=0

3. In GULINE when KSHOW=2:

   a. The line is completely outside the frame.

   b. The line is completely within the frame.

   c. The line is partially within the frame and has been scissored.

4. The name of the routine which displays a numeric font is:

   a. GFONTA

   b. GFONTN

   c. FONT

5. If a program requests 20 characters (two 10-character words) of input and only three are light-pen picked from the font or typed from the keyboard:

   a. The second word will be blank-filled.

   b. The first word will be blank-filled and the second word will have unpredictable contents.

   c. Neither word will be blank-filled.

CODING PROBLEMS

Assume masks are set:

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker
NCON	=	1
MBYTE	=	310

1. Display an alphanumeric font with center at (600, 600).

2. Erase the alphanumeric font created in Problem 1.

3. X and Y axes are displayed on the screen and we are calculating and plotting least squares fit lines through a set of points. The endpoints of the line have coordinates AX, AY and BX, BY. Before displaying the line, write the call that determines the points at which the line intersects the plot frame if it intersects the frame. The frame has center (750, 750) and upper right-hand corner (1000, 1000).

4. Use GULINE to scissor a given line in a 14 by 14-inch square with center (0, 0). Display the scissored line as an ignore item.

5. Use GUARC to scissor a circle with radius IRAD within a 14 by 14-inch square with center at (0, 0). Display the appropriate axes as ignore items.

6. Frame-scissor an arc drawn on the grid of Problem 5. The center of the arc is at (0, 0) with endpoints at H1, V1 and H2, V2.

7. a. Display a light register of 10 characters with the left end of the underline at (-1000, 700).

   b. Use GIEOM to designate ";" as the end-of-message character.

   c. Return up to nine characters and edit out the EOM character.

   d. Test for at least one character other than the EOM character.

8. Display the characters returning in Problem 7 on the console beginning at (-35, 0).

9. Suppose that the graphics console is equipped with a function keyboard. Use GIKYBD to associate an ID block with the function keyboard of graphics console 1.

10. Supply the application executive (AE) call which will be used to call a routine called CUBE from OVERLAY(1, 0).

ADDITIONAL CODING EXAMPLES A

This Appendix contains examples of code for creating light buttons, lines, circles, arcs, figures, and fonts. It also includes several complete programs. For most of the coding examples, assume that dimensioning has been performed; special cases will include dimension statements.

CREATE A LIGHT BUTTON

```
NBYTE = 0
NAME = 8HDISPLAY
CALL GURSET (-1500,0,102B,IBUF,NBYTE,MBYTE)
CALL GUAN (NAME,8,IBUF,NBYTE,MBYTE)
CALL GIDISP (NCON,IBUF,NBYTE,IDDAD,1,1)
```

The characters to be displayed are DISPLAY. GURSET is called to generate a reset sequence byte-stream which is placed in temporary user buffer IBUF. GUAN is called to generate the byte-stream which will cause the alphanumerics to be displayed. This byte-stream follows the reset sequence byte-stream in IBUF. Parameter NBYTE is automatically updated to reflect the number of bytes in IBUF.

GIDISP is called to send a copy of the contents (NBYTE bytes) of IBUF to the 1700 series buffer translator. NBYTE is then set back to zero by the application program to initialize IBUF for the next byte-stream. The buffer translator calls the 1700 series graphics BGP which generates a display byte-stream and places it in the display controller. Once the byte-stream is in the display buffer, it is displayed. Note that parameters IDDT and IDDC in the call to GIDISP are both one. For IDDT=1 to be meaningful to the system (i.e., when the button is picked with the light-pen by the operator, the system interprets IDDT for the action it is to perform), the call

```
CALL GIMASK (NCON, 0, 1, 16+8)
```

must be executed prior to the pick of the button. This call defines items with the IDDT designation of 1 as buttons. The buttons so designated will have the marker mask set such that when the button is picked, it will blink (assuming that the button is not already blinking).

There are several means by which alphanumeric input can be provided. The above example uses a dimensioned array. A data statement could be used instead, as in this example:

```
COMMON/DATA/NAME(2)
DATA NAME(I,I = 1,2)/8HDISPLAY,6HBUFFER)
```

An ENCODE and FORMAT statement could also be used to provide alphanumeric input.

```
        ENCODE (16,1,NAME)
        FORMAT (16HDISPLAY   BUFFER)
```

CREATE A LIGHT BUTTON—UNDER APPLICATION EXECUTIVE

```
        NBYTE = 0
        NAME = 8HDISPLAY
        NAME(2) = 4RDISP
        CALL GURSET (-1500,0,102B,IBUF,NBYTE,MBYTE)
        CALL GUAN (NAME,8,IBUF,NBYTE,MBYTE)
        CALL GIDISP (NCON,IBUF,NBYTE,IDDAD,1,1,NAME(2))
```

Note that the calls are essentially the same as those in the previous example. The only difference is that in this call to GIDISP, an additional parameter is used to identify the task to be called by the application executive when the button is picked. To relate a task to the light button that initiates a branch to that task, the first four characters of the light button name are used as the task name; thus, if the button DISPLAY is picked and there has been a call to AETSKR, the application executive brings task DISP into core from mass storage for execution.

GENERATE AND DISPLAY A LINE—NOT FRAME-SCISSORED

```
        NBYTE = 0
        CALL GURSET (-1000,0,102B,IBUF,NBYTE,MBYTE)
        CALL GUSEGS (-1000,0,1000,0,1,0,IBUF,NBYTE,MBYTE)
        CALL GIDISP (NCON,IBUF,NBYTE,IDDAD,2,1,0,0)
```

These calls will display a line from (-1000, 0) to (1000, 0). The line is solid, light-pen sensitive, and does not blink.

A previous call to GIMASK was made for type 2 as follows:

```
        CALL GIMASK (NCON, 0, 2, 16+4)
```

This call associates the marker mask with type 2 and makes all type 2 display items string pick items.

Note that all ID block parameters are used.

The user may wish to store items such as the item display address or bead address in the ID block. If, for instance, the display address for the line is in IDDAD, GIMOVE can be used to insert this display address into the ID block as follows:

```
CALL GIMOVE (-0, -0, -0, IDDAD, 2, 1, 0, IDDAD)
```

The display address is now in the IDWB word of the ID block for that line. The display address can be extracted from the block by GIFSID any time the line is picked. It is important to have access to the display addresses of the items if they are to be erased, copied, or have their ID blocks modified (as was the case above in the call to GIMOVE).

If the data handler is used and a bead is formed for this line, the bead address could be placed in IDWA. It is usually more convenient to put the display address in the bead rather than in the ID block, but these decisions are up to the individual.

GENERATE AND DISPLAY A FRAME-SCISSORED LINE

```
      NBYTE = 0
      CALL GULINE (0,0,1000,1000,500.,0.,-600.,-500.,KSHOW,IHS,IVS,IHF,IVF)
      CALL GURSET (IHS,IVS,102B,IBUF,NBYTE,MBYTE)
      CALL GUSEGS (IHS,IVS,IHF,IVF,1,0,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,IDISPAD,2,2,0,0)
```

GULINE is called to set up the size and position of the frame for frame-scissoring. Floating-point coordinates for the beginning and end of the line are given in the call and converted by the system to fixed-point coordinates. Although not shown in this example, it is advisable to test KSHOW for equality to zero to check whether the described line can be displayed. If KSHOW=0, there is no reason to call GURSET, GUSEGS, and GIDISP.

The following code will perform the check and skip to statement 2 if the line cannot be displayed:

```
      NBYTE = 0
      CALL GULINE (0,0,1000,1000,500.,0.,-600.,-500.,KSHOW,IHS,IVS,IHF,IVF)
      IF (KSHOW .EQ. 0)GO TO 2
      CALL GURSET (IHS,IVS,102B,IBUF,NBYTE,MBYTE)
      CALL GUSEGS (IHS,IVS,IHF,IVF,1,0,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,IDISPAD,2,2,0,0)
    2 CONTINUE
```

GENERATE AND DISPLAY A CIRCLE—NOT FRAME-SCISSORED

```
      NBYTE = 0
      CALL GURSET (0,0,102B,IBUF,NBYTE,MBYTE)
      CALL GUARCG (1,0,0, 300,0,300,0,-0,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,ICRCDSPD,2,3,0,0)
```

This code generates a circle with solid line style. The center of the circle is displaced from the reset coordinates (0, 0) by 300 display grid units to the left because circles and arcs are displayed in a counterclockwise manner from the initial point to the terminal point. Thus, the center of this circle is at (-300, 0).

The reader should note that since all of the circle can be displayed on the display surface, parameter KSHOW is set to 1; the arc-scissoring routine GUARC was not used to make that determination. An example of arc-scissoring is given in the next paragraph.

GENERATE AND DISPLAY A FRAME-SCISSORED CIRCLE

```
        NBYTE = 0
        DIMENSION IHS(5),IVS(5),IHF(5),IVF(5)
        CALL GUARC (0,0,1000,1000,0.,0.,1125.,0.,1125.,0.,KSHOW,IHC,IVC,IHS,IVS,
        IHF,IVF)
        IF (KSHOW .EQ. 0)GO TO 3
        CALL GURSET (IHS,IVS,102B,IBUF,NBYTE,MBYTE)
        CALL GUARCG (KSHOW,IHC,IVC,IHS,IVS,IHF,IVF,-0,IBUF,NBYTE,MBYTE)
        CALL GIDISP (NCON,IBUF,NBYTE,ICRCAD,2,4,0,0)
      3 CONTINUE
```

Since it is possible to have a circle scissored into four segments and an arc into five segments, arrays had to be dimensioned to accept the starting and ending points of the arc segments. The coordinates used in the call to GUARC are deliberately chosen to generate four arc segments for the subject circle. Since it is known in advance that four arc segments will be displayed, KSHOW is examined here for equality to zero as a formality; however, this is not often the case and the test for KSHOW should be made as a matter of good programming practice.

The remainder of the code merely resets the CRT beam to the start of the first arc segment. GUARCG is called to generate the four arc segments, and GIDISP transfers the byte-stream from IBUF to the display buffer.

GENERATE AND DISPLAY A 2-INCH SQUARE—NOT FRAME-SCISSORED

```
        NBYTE = 0
        CALL GURSET (0,0,102B,IBUF,NBYTE,MBYTE)
        CALL GUSEGS (0,0,400,0,1,0,IBUF,NBYTE,MBYTE)
        CALL GUSEG (400,-400,1)
        CALL GUSEG (0,-400,1)
        CALL GUSEG (0,0,1)
        CALL GIDISP (NCON,IBUF,NBYTE,ISQDSPAD,2,6,0,0)
```

This square starts at the origin of the scope display grid. The first line segment goes right 2 inches, the second segment goes down 2 inches, the third goes left 2 inches, and the last is drawn up 2 inches to complete the square.

GENERATE AND DISPLAY A COLUMN OF FIVE HORIZONTAL LINES

```
NBYTE = 0
CALL GURSET (0,0,102B,IBUF,NBYTE,MBYTE)
CALL GUSEGS (0,0,400,0,1,-0,IBUF,NBYTE,MBYTE)
CALL GUSEG (0,-100,0)
CALL GUSEG (400,-100,1)
CALL GUSEG (0,-200,0)
CALL GUSEG (400,-200,1)
CALL GUSEG (0,-300,0)
CALL GUSEG (400,-300,1)
CALL GUSEG (0,-400,0)
CALL GUSEG (400,-400,1)
CALL GIDISP (NCON,IBUF,NBYTE,ILNDSPAD,2,7,0,0)
```

The first line is drawn from the scope display grid origin to a point 2 inches to the right. The first call to GUSEG positions the beam for the next line. Note that the GUSEG call turns the beam off and nothing is displayed. The second call to GUSEG generates the byte-stream for the second of the five lines to be displayed. From this point it is a repetition of the first two calls to GUSEG until the entire byte-stream is generated. GIDISP is then called to transfer the byte-stream to the display buffer for subsequent display.

An alternate method for generating the same five lines uses a call to GUSEGA.

```
DIMENSION IH(10),IV(10),IBEAM(10)
IH(1)=IH(3)= IH(5)=IH(7)=IH(9)=IV(1)=IV(2)=0
IH(2)=IH(4)=IH(6)=IH(8)=IH(10)=400
IV(3)=IV(4)=-100
IV(5)=IV(6)=-200
IV(7)=IV(8)=-300
IV(9)=IV(10)=-400
IBEAM(1)=IBEAM(3)=IBEAM(5)=IBEAM(7)=IBEAM(9)=1
IBEAM(2)=IBEAM(4)=IBEAM(6)=IBEAM(8)=IBEAM(10)=0
NBYTE = 0
CALL GURSET (IH,IV,102B,IBUF,NBYTE,MBYTE)
CALL GUSEGA (IH,IV,IBEAM,9,-0,IBUF,NBYTE,MBYTE)
CALL GIDISP (NCON,IBUF,NBYTE,ILNDSPAD,2,7,0,0)
```

In this example, the single call to GUSEGA generates the byte-streams for all the line segments by referring to the IH, IV, and IBEAM arrays.

GENERATE AND DISPLAY A LINE AS A MACRO

```
NBYTE = 0
CALL GUSEGS (0,0,400,500,1,-0,IBUF,NBYTE,MBYTE)
CALL GIMAC (NCON,IBUF,NBYTE,MAD)
```

The above code generates the byte-stream for the line, transfers a copy of the byte-stream to the fixed address area of the display buffer, and returns a macro address in MAD. The line is not displayed at this time. To display the line, the following code is required.

```
NBYTE = 0
CALL GURSET (100,200,102B,IBUF,NBYTE,MBYTE)
CALL GUMACG (MAD,1,IBUF,NBYTE,MBYTE)
CALL GIDISP (NCON,IBUF,NBYTE,MACDSPAD,2,8,0,0)
```

The call to GURSET determines where the macro is displayed.

DISPLAY AN ID BLOCK RETURNED FROM A CALL TO GIFSID

GIFSID is used to return the ID blocks of string pick items in the FETCH queue. This example shows the use of GIFSID in conjunction with calls to display the contents of the ID block. Assume that the proper calls to GIMASK have been made, the console operator has picked at least one string pick item and a button, and that this routine was entered after a GIBUT call or through an AETSKC or AETSKR call.

```
      DIMENSION IBCD (20)
      N = 1
      NBYTE = 0
      CALL GIFSID(NCON,1,IT,IC,IA,IB,IH,IV)
      CALL GURSET (-1200,1300,102B,IBUF,NBYTE,MBYTE)
      ENCODE (49,30,IBCD) NCON,IT,IC,IA,IB,IH,IV
   30 FORMAT (7HGIFSID(,3(I5,1H,) ,2(1X,R4,1H,),I5,1H,I5,1H))
      CALL GUAN(IBCD,49,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,IDMESS,2,0,0,0)
```

GIFSID is called to extract one string pick ID block and return it to IT, IC, IA, IH, and IV. GURSET is used to set the point at which the A/N display starts. The ENCODE and FORMAT statements assemble the indicated characters into array IBCD, which GUAN uses as input to generate the byte-stream for the display. GIDISP transfers the byte-stream to the display buffer and appends the indicated ID block to the byte-stream. This display consists of the call to GIFSID with its calling and result parameters.

MOVE A DISPLAY ITEM

Assume that a line has been displayed as a display item and has its display address stored in IDSPAD(12). Further, it is required that the line be displayed at new coordinates IH=1000, IV=-400. It is coded as

```
CALL GIMOVE (1000, -400, -0, IDSPAD(12))
```

The ICODE value, -0, indicates that the ICODE already associated with the line is not to be changed. The call is truncated after IDSPAD(12), since the ID block for the line is also unchanged.

The line could have been moved and its ID block changed by including the new ID parameters to replace the existing ones. Remember that the ID block cannot be expanded beyond the size already in existence for the line. For instance, if only parameters IDDT and IDDC were used in the call to GIDISP for this line, GIMOVE cannot be used to add ID parameters, such as IDWA or IDWB, since space has not been allocated for these parameters.

COPY A DISPLAY ITEM

This is similar to using GIMOVE except that a copy is moved and the original still exists. To copy the line described in the last example, use the code

```
        CALL GICOPY (IDSPAD(12), NCON, 1000, 400, 106B, ICPYADD, IT, IC, IA, IB)
```

A copy of the line with display address IDSPAD(12) will be displayed 4 inches higher on the scope and the copy will blink. The display address of the copy will be put in ICPYADD.

ERASE A DISPLAY ITEM

To erase the line made as a result of the call GICOPY that was just described, use the code

```
        CALL GIERAS(ICPYADD)
```

ERASE A MACRO

The programmer may recall that a macro is displayed in two steps:

1. Generate the byte-stream and call GIMAC to put the byte-stream in the fixed-address area of the display buffer. The address of the macro is returned in an output parameter referred to as MAD.
2. Generate a reset sequence with a call to GURSET; call GUMACG giving the MAD parameter. GUMACG will generate a calling sequence for that macro. Call GIDISP to transfer the byte-stream generated in this second step to the display buffer and the item is displayed.

He now wishes to erase the macro from both the fixed- and floating-address areas of the display buffer. The procedure is inflexible because the calling sequence in the floating address area must be erased first, then the display item byte-stream from the fixed-address area. If the calling sequence remains while the macro is erased, the display jumps to a non-existent macro and the result is chaos. The following code will correctly erase a macro:

```
CALL GIERAS (ISBD)
CALL GIMACE (MADD)
```

ISBD	Display address in the floating-address area for the calling sequence
MADD	Macro address in the fixed-address area

To erase a specific display of a macro and still retain its byte-stream in the final address area, call GIERAS and not GIMACE. When a macro is erased with GIERAS, the area in the display buffer is not contracted.

CREATING ALPHANUMERIC DISPLAY FONTS

For certain applications, the programmer may wish to provide the console user with a display font other than the two supplied in the 6000 series Basic Graphics Package (GFONTA and GFONTN). The following discussion covers some of the more important points that a programmer should consider when creating his own display font.

FONT CHARACTER RECOGNITION

The 1700 series Basic Graphics Package recognizes a sequence of display generation bytes followed by a one-word ID as a display font character. When the character is picked with the light-pen and GIANS has been called, the ID word is queued on an alphanumeric string so that it can be sent to the application program when a GIANE call occurs. Each 8-bit ID word in the 1700 package is an ASCII character and is converted to 6000 package display code before being sent to the 6000 series computer application program.

Because of this processing, the application programmer can create font characters by supplying the one-word ASCII ID through a call to GUBYTE. The general form for GUBYTE is

```
CALL GUBYTE (IBYTE, L, IBUF, NBYTE, MBYTE)
```

IBYTE is the octal equivalent of the hexadecimal internal code for alphanumeric characters or the first word of an array. L is the number of consecutive words in IBYTE from which bytes are to be transferred. For example, the three calls

```
CALL GURSET(IH,IV,ICODE,IBUF,NBYTE,MBYTE)
CALL GUAN(1LA,1,IBUF,NBYTE,MBYTE)
CALL GUBYTE(101B,1,IBUF,NBYTE,MBYTE)
```

create an alphanumeric font of one character, A, at screen coordinates IH and IV. The ASCII code for A is 101_8.

The call

```
        CALL GIDISP(NCON, IBUF, NBYTE, IDDAD, -0)
```

then displays this one character font. After the font appears on the screen, the call

```
        CALL GIANS (NCON, 10, IH1, IV1)
```

creates a light register at screen coordinates IH1 and IV1; this register can contain up to 10 of the As, if the character is picked that many times.

If the character A is picked once and GIANE is called, the parameters returned to the call will be:

```
        NC   = 1
        IBCD = Abbbbbbbbb
```

where the letter b indicates a blank.

SPECIAL CHARACTERS

Two special characters are defined for the 1700 package. These two characters, backspace and clear, allow the console operator to remove characters which have been queued since the call to GIANS and before the next call to GIANE occurs.

BACKSPACE

Any display followed by a one-word ID of 137B is defined as a backspace character. When such a character is picked with the light-pen, the last picked character in the light register is erased from the display and the underline is restored; the ID of the erased character is also removed from the buffer or queued alphanumeric information.

CLEAR

Any display followed by a one-word ID or 177B is defined as a clear character. When such a character is picked with the light-pen, all of the characters currently in the light register are erased and the entire underline is restored; in addition, the IDs for all of the erased characters are removed from the buffer of queued alphanumeric information.

Backspace and clear have no other effect on alphanumeric picking.

RESET SEQUENCES

When a GURSET call is used in the definition of a font character, the ICODE s bit (bit 2^6) must be set. The s bit of the reset sequence is the enable light-pen bit; if it is not set, the character's ID word is not read when a pick is made, and the character consequently cannot be entered into the light register or queued for 6000 series computer processing.

A font character can be generated without a reset sequence by using a GUAN call with NC set equal to one, but a no-operation instruction must precede the GUAN call in the character's IBUF. This no-operation may be supplied by a GUBYTE call of one byte, where the byte is a positive zero value.

CONSERVING ID WORD SPACE

The ID words IDDT, IDDC, IDWA, and IDWB of the GIDISP call or calls which display font characters need not be referenced; the 1744 Buffer space they normally occupy can be conserved by truncating the parameter list with a closing or right parenthesis after IDDAD.

DYNAMIC ADDITION OF CHARACTERS

Characters may be added to an existing console display font by successive calls to GIDISP at any time. Duplicates of the same character, i.e., characters with the same ASCII code ID words, may be present in a font.

SAMPLE FONT CREATION ROUTINES

The following subroutine creates a display font containing

```
      0 1 2 3 4 5 6 7 8 9 X

      SUBROUTINE NFONT (NCON,IBUF,NBYTE,MBYTE,IDDAD)
      DIMENSION IBCD(10)
      DATA (IBCD(I),I=1,10)/1L0,1L1,1L2,1L3,1L4,1L5,1L6,1L7,1L8,1L9/
      CALL GURSET (0,-600,103B,IBUF,NBYTE,310)
      ICON1 = 60B
      ICON2 = 71B
      DO 1 I = ICON1,ICON2
      J = I -57B
      CALL GUBYTE(0,1,IBUF,NBYTE,MBYTE)
C     THE PRECEDING CALL PROVIDES A NO-OP BEFORE EACH GUAN CALL TO
C     GENERATE A CHARACTER AND IS NECESSARY ONLY WHEN EACH CHARACTER
C     IS GENERATED BY A SEPARATE GUAN CALL
      CALL GUAN (IBCD(J), 1, IBUF, NBYTE, MBYTE)
C     THE PRECEDING CALL GENERATES ONE OF THE FONT CHARACTERS
      CALL GUBYTE (I, 1, IBUF, NBYTE, MBYTE)
C     THE FOLLOWING CALL PROVIDES SPACING BETWEEN CHARACTERS AND
C     COULD BE REPLACED BY A GURSET CALL
1     CALL GUAN (1L ,1, IBUF, NBYTE, MBYTE)
      CALL GUAN (2L  , 2, IBUF, NBYTE, MBYTE)
      CALL GUBYTE (0, 1, IBUF, NBYTE, MBYTE)
      CALL GUAN (1LX, 1, IBUF, NBYTE, MBYTE)
      CALL GUBYTE (130B, 1, IBUF, NBYTE, MBYTE)
```

```
C  THE THREE PRECEDING CALLS CREATE AND IDENTIFY THE CHARACTER X
C  AS AN END-OF-MESSAGE CHARACTER FOR USE IN GIEOM ASSIGNMENT
C  THE FOLLOWING CALL DISPLAYS THE FONT
   CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, -0)
   RETURN
   END
```

The programmer can also create display font characters of any size he wishes; he need not use the size characters that are defined by the 1700 Basic Graphics Package alphanumeric macros. For example, the three following calls create a circle with a center at IHC and IVC, and an initial/termination point at IH and IV. This circle is queued as an alphanumeric O when picked with the light-pen.

```
CALL GURSET (IH,IV,ICODE,IBUF,NBYTE,MBYTE)
CALL GUARCG (1,IHC,IVC,IH,IV,IH,IV,IBUF,NBYTE,MBYTE)
CALL GUBYTE (117B,1,IBUF,NBYTE,MBYTE)
```

Note that the ASCII code equivalent of O is 117B ($4F_{16}$).

The programmer can create a true/false font with coding like the following:

```
   CALL GURSET (IH1, IV1, ICODE, IBUF, NBYTE, MBYTE)
   CALL GUAN (4HTRUE, 4, IBUF, NBYTE, MBYTE)
   CALL GUBYTE (124B, 1, IBUF, NBYTE, MBYTE)
C  THE PRECEDING CALLS CREATE THE WORD TRUE BEGINNING AT IH1/IV1
C  AND QUEUE AN ALPHANUMERIC T (=124B) WHEN IT IS PICKED
   CALL GURSET (IH2, IV2, ICODE, IBUF, NBYTE, MBYTE)
   CALL GUAN (5HFALSE, 5, IBUF, NBYTE, MBYTE)
   CALL GUBYTE (106B, 1, IBUF, NBYTE, MBYTE)
C  THE PRECEDING 3 CALLS CREATE THE WORD FALSE BEGINNING AT IH2/IV2
C  AND QUEUE AN ALPHANUMERIC F (=106B) WHEN IT IS PICKED
```

STAR OF DAVID PROGRAM †

The following printout lists all of the cards needed for the file creation and execution runs of a simple graphics job. This job consists of one short primary overlay that draws a star and a square at one console, then creates two light buttons. The console user is informed that the square is supposed to be within the star; he then picks the proper button to center the square within the star, and the figure is moved. If the button he picks is invalid, he receives a message and the job aborts.

†See page A-61 for a different approach to the Star of David problem.

CARD SEQUENCE FOR GRAPHICS FILE CREATION

```
XMPL,P37,T1000,CM60000.
RUN(S)
LGO.
AEFILE.
7
8
9

    OVERLAY(SCR,0,0)
    PROGRAM CREATE
    CALL MAIN
    STOP
    END

      OVERLAY(1,0)
      PROGRAM LITTLE
      DIMENSION IBUF(64),IBCD(13),IBB(13)
      DATA IBCD(1)/40HSUPPOSED TO BE A SQUARE INSIDE A STAR    /
      DATA(IBCD(I),I=5,11,2)/20HMOVE SQUARE LEFT    ,
     120HMOVE SQUARE UP      ,20HMOVE SQUARE DOWN    ,
     120HMOVE SQUARE RIGHT   /
      DATA IBCD(13)/10HWRONG IDDT/
      DATA(IBB(K),K=1,9,2)/20HRELEASE             ,20HCHECK
     1    ,20HRESTART             ,20HVERY GOOD           ,
     120HTRY AGAIN           /
      NCON=9
C  CONNECT CONSOLE
      CALL GICNJB(NCON)
C  SET ITEM MASKS
      DO 10 I=2,5
      IDDTS=2**(I-1)
      IMASK=IDDTS
10    CALL GIMASK(NCON,-0,IDDTS,IMASK)
C   SET DISPLAY CONSTANTS
      ICODE=103B
      ISTYLE=7777B
      MBYTE=310
      NBYTE=0
C   START DRAWING SQUARE
      CALL GURSET(-1400,0,3B,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(-1400,0,ISTYLE,IBUF,NBYTE,MBYTE)
      CALL GUSEG(-1200,200,0)
      CALL GUSEG(-1200,-200,1)
      CALL GUSEG(-1600,-200,1)
      CALL GUSEG(-1600,200,1)
      CALL GUSEG(-1200,200,1)
C   DISPLAY SQUARE AS A SINGLE PICK ITEM
      CALL GIDISP(NCON,IBUF,NBYTE,IDRSV,2,1)
C   DRAW STAR OF DAVID (TWO TRIANGLES)
      CALL GURSET(-500,300,ICODE,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(-500,300,ISTYLE,IBUF,NBYTE,MBYTE)
      CALL GUSEG(500,300,1)
      CALL GUSEG(0,-600,1)
      CALL GUSEG(-500,300,1)
      CALL GUSEG(-500,-300,0)
```

```
C    DRAW SECOND TRIANGLE
      CALL GUSEG(500,-300,1)
      CALL GUSEG(0,600,1)
      CALL GUSEG(-500,-300,1)
C    DISPLAY STAR AS STRING PICK WITH MARKER MASK SET
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,24B,2)
C   LABEL THE FIGURES
      CALL GURSET(-744,1400,3B,IBUF,NBYTE,MBYTE)
      CALL GUAN(62HMOVE THE SQUARE INTO THE STAR AND CHECK IF IT IS IN T
     1HE CENTER,62,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,1,3)
C     MAKE FOUR BUTTON CHOICES
      DO 20 J=5,11,2
      IF(J.EQ.5)IH=-1200
      IF(J.EQ.7)IH=-600
      IF(J.EQ.9)IH=0
      IF(J.EQ.11)IH=600
      CALL GURSET(IH,-900,ICODE,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD(J),20,IBUF,NBYTE,MBYTE)
20    CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,8,J)
      DO 52 N=13,21,2
      IDDT=8
      IV=-1200
      IF(N.EQ.13)IH=-105
      IF(N.EQ.13)IV=1800
      IF(N.EQ.15)IH=-96
      IF(N.EQ.17)IH=96
      IF(N.LT.19)GO TO 26
      IH=-175
      IV=1200
      ICODE=1B
      IDDT=1
26    M=N-12
      CALL GURSET(IH, IV  ,ICODE,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBB(M),9,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,IDDT,N)
      M=0
      IH=-1400
      IV=0
      IF(N.EQ.19)IGOOD=IDDAD
      IF(N.EQ.21)ITRY=IDDAD
52    CONTINUE
      CALL GITCON(NCON,IH,IV)
      CALL GITIMV(NCON,IDRSV)
36    CALL GIBUT(0,NCON,IDDT,IDDC)
      IF(IDDC.LT.19.AND.IDDC.GT.3)GO TO 43
      CALL GIABRT(NCON,10HWRONG PICK,10)
      STOP
43    L=(IDDC-3 )/2
      CALL GITCOF(NCON,IH,IV)
      GO TO(44,45,46,47,48,49,60)L
44    IH=IH-50
      GO TO 50
45    IV=IV+50
      GO TO 50
46    IV=IV-50
      GO TO 50
47    IH=IH+50
```

```
50     CALL GIMOVE(IH,IV,103B,IDRSV)
       CALL GITCON(NCON,IH,IV)
       IF(M) 55,36,55
55     CALL GIMOVE(-2040,2040,1B,IGOOD)
       CALL GIMOVE(-2040,-2040,1B,ITRY)
       M=0
       GO TO 36
49     M=1
       CALL GITCON(NCON,IH,IV)
       IF(IABS(IH).LT.16.AND.IABS(IV).LT.16)GO TO 62
       CALL GIMOVE(-108,1700,7B,ITRY)
       CALL GIMOVE(-2040,-2040,1B,IGOOD)
       GO TO 36
62     CALL GIMOVE(-108,1200,7B,IGOOD)
       CALL GIMOVE(-2040,-2040,1B,ITRY)
       GO TO 36
60     IH=-1400
       IV=0
       GO TO 50
48     CALL GIABRT(NCON,12HJOB RELEASED,12)

C    RELEASE CONSOLE
     CALL GICNRL(NCON)
     STOP
     END
7
8
9
```

Two file names cause MAIN to recognize the file creation job:

```
XMPL
SCR
6
7
8
9
```

CARD SEQUENCE FOR GRAPHICS EXECUTION JOB

```
LPMX, P37, T200, CM60000.
COMMON, XMPL.
RUN(S)
LGO.
RELEASE, XMPL.
EXIT.
RELEASE, XMPL.
7
8
9
    OVERLAY(SCR, 0, 0)
    PROGRAM EXECUTE
    CALL MAIN
    STOP
    END
7
8
9
```

One file name causes MAIN to recognize the file execution job:

```
XMPL
6
7
8
9
```

PROGRAM CLASDEM

The example given below consists of a program which sets the ID processor mask, generates a line and a circle with subroutines, and generates four light buttons: one to display a line, one to display a circle, one to erase a picked line or circle, and one to terminate the application. Coordinate information for display item position will be furnished by calls to the tracking cross position fetch routine GITCOF.

The console number, NCON, is input from a card and stored in common location NCON. The program is designed to terminate automatically when 10 components have been created.

The program starts by reading a card for the console number and storing it in common location NCON. Console NCON is assigned to the job with a call to GICNJB.

The byte-stream for the line is generated and stored in IBUF with a call to GUSEGS; it is made a macro and stored in the macro area of the display buffer with a call to GIMAC. The macro address is returned in MAD(1)

The byte-stream for the circle is generated and stored in IBUF with a call to GUARCG; it is made a macro and stored in the macro area of the display buffer with a call to GIMAC. The macro address is returned in MAD(2). The macro addresses are needed when the line or circle is to be displayed. The MAD parameters are used in calls to GUMACG which generates the macro call. The macro call provides access to the line and circle byte-streams for display.

Once the byte-streams are taken care of, the next step is to set up the ID processor mask for the light buttons and components. This is done with calls to GIMASK. Light buttons are designated type 1 and set to blink when picked as operator feedback. Lines and circles designated type 2 are single pick items set to blink when picked. This means that if more than one line or circle is picked for erasure, only the last one picked has its ID block retained; the preceding ID blocks are deleted.

The next step is to create and display four light buttons:

- LINE
- CRCL

- ERAS
- OVER

The first three light buttons call subroutines LINE, CIRCLE, and ERASE, respectively. OVER calls GICNRL to terminate the application.

The user at this time sees the light buttons displayed along with the tracking cross. He responds by picking up the cross with the light-pen and moves the cross to where he wants a line or circle to be displayed. His next step is to select either the LINE or CRCL light button to indicate whether he wants a line or circle to be displayed at the tracking cross position.

GIBUT has been called and is waiting for a light button pick. Once the user has responded, the ID block of the selected light button is returned; a computed GO TO is executed, based on the IDDC parameter indicating the particular light button selected. It is safe to assume that either LINE or CRCL was selected since there is nothing to erase and it is not likely that the operator would terminate at this time. The following paragraphs analyze the functions of subroutines LINE, CIRCLE, and ERASE which are called as a result of the execution of the computed GO TO.

SUBROUTINE LINE

LINE retrieves the horizontal and vertical coordinates of the tracking cross with a call to GITCOF. The coordinates are returned in ITH and ITV. A reset sequence is created using ITH and ITV for the position at which the line will start. This is followed with a call to GUMACG to generate the macro call, which provides access to the line byte-stream in the macro area of buffer memory. Note that the macro address given in the call is the one for the line (MAD(1)).

The bytes for the reset sequence and the subroutine call are temporarily stored in IBUF. Now GIDISP is called to transfer NBYTE bytes of IBUF to the display item area of buffer memory. The display address is stored in IDDAD(K + 1) to enable the operator to erase the line if he desires. Note that the ID block contains the line type (2), the line code (1), and K + 1. K + 1 will be used in subroutine ERASE to determine which particular display item is to be erased. K is then incremented to be ready for the next item to be displayed. K is also tested for equality to 11 to see if all 10 locations of IDDAD have been used. If not, the program continues. If all the locations have been used, GICNRL is called to terminate the application.

SUBROUTINE CIRCLE

CIRCLE is identical to LINE except that MAD(2) is used instead of MAD(1) and the circle code (2) is used in the call to GIDISP instead of code (1) (as was the case with LINE). Both LINE and CIRCLE could easily be combined into one routine; however, the redundancy reinforces the learning process.

SUBROUTINE ERASE

ERASE fetches the ID block of the line or circle picked by the operator for erasure. IDWA contains the K parameter set into the ID block by GIDISP when the subroutine call sequence was generated. GIERAS is then called to erase the line or circle whose display address is found in IDDAD(IDWA), where IDWA once again is the K parameter. This does not remove the byte-stream from the macro area; however, it is referenced from the display item area. A printout of these routines is shown below.

```
                   OVERLAY (SCR,0,0)
                   PROGRAM M(INPUT,OUTPUT)
000003             COMMON IBUF (100), MAD (2), IDDAD (10), NBYTE, MBYTE, NCON, K
000003             CALL MAIN
000004             END

                   OVERLAY (1,0)



                   PROGRAM CLASDEM
000003             COMMON IBUF (100), MAD (2), IDDAD (10), NBYTE, MBYTE, NCON, K
             C
             C        MAD(1)    LINE MACRO ADDRESS
             C        MAD(2)    CIRCLE MACRO ADDRESS
             C
             C        IDDAD(1) TO IDDAD(10)  DISPLAY ITEM ADDRESS
             C
             C        DISPLAY ITEM BLOCK
             C            WORD1  DISPLAY TYPE
             C                   1= BUTTON
             C                   2= SINGLE PICK
             C            WORD2  DISPLAY ITEM
             C                   1= LINE
             C                   2= CIRCLE
             C            WORD3  DISPLAY ITEM MATRIX ADDRESS
             C
             C     SIGN ON CONSOLE
000003             READ 1, NCON
000011           1 FORMAT (I2)
000011             CALL GICNJB (NCON)
000013             MBYTE = 310
000014             NBYTE = 0
000015             K = 0
```

```
          C      GENERATE LINE MACRO
000016           CALL GUSEGS (0, 0, 600, 0, 1, -0, IBUF, NBYTE, MBYTE)
000026           CALL GIMAC(NCON,IBUF,NBYTE,MAD(1))
000031           NBYTE = 0
          C      GENERATE CIRCLE MACRO
000032           CALL GUARCG (1,0,0,300, 0, 300, 0, -0, IBUF, NBYTE, MBYTE)
000045           CALL GIMAC(NCON,IBUF,NBYTE,MAD(2))
000050           NBYTE = 0
          C      SET BUTTON MASK
000051           CALL GIMASK (NCON,-0,1,16+8)
          C      SET SINGLE PICK MASK
000056           CALL GIMASK (NCON,-0,2,16+2)
          C      DISPLAY LINE BUTTON
000063           CALL GURSET (0, -1500, 102B, IBUF, NBYTE, MBYTE)
000067           CALL GUAN (4HLINE, 4, IBUF, NBYTE, MBYTE)
000073           CALL GIDISP (NCON, IBUF, NBYTE, IDA, 1, 1)
000077           NBYTE = 0
          C      DISPLAY CIRCLE BUTTON
000100           CALL GURSET (0, -1600, 102B, IBUF, NBYTE, MBYTE)
000104           CALL GUAN (4HCRCL, 4, IBUF, NBYTE, MBYTE)
000110           CALL GIDISP (NCON, IBUF, NBYTE, IDA, 1, 2)
000114           NBYTE = 0
          C      DISPLAY ERASE BUTTON
000115           CALL GURSET (0, -1700,  102B, IBUF, NBYTE, MBYTE)
000121           CALL GUAN (4HERAS, 4, IBUF, NBYTE, MBYTE)
000125           CALL GIDISP (NCON, IBUF, NBYTE, IDA, 1, 3)
000131           NBYTE = 0
          C      DISPLAY OVER BUTTON
000132           CALL GURSET (0, -1800, 102B, IBUF, NBYTE, MBYTE)
000136           CALL GUAN (4HOVER, 4, IBUF, NBYTE, MBYTE)



000142           CALL GIDISP (NCON, IBUF, NBYTE, IDA, 1, 4)
000146           NBYTE = 0
          C      TURN ON TRACKING CROSS
000147         2 CALL GITCON (NCON,0,0)
          C      WAIT TO PICK BUTTON
000152           CALL GIBUT (0,NCON,IDDT,IDDC)
000155           GO TO (3, 4, 5, 6), IDDC
000165         3 CALL LINE
000166           GO TO 2
000167         4 CALL CIRCLE
000170           GO TO 2
000171         5 CALL ERASE
000172           GO TO 2
          C      JOB DONE, RELEASE CONSOLE
000173         6 CALL GICNRL (NCON)
000175           END
```

```
                   SUBROUTINE CIRCLE
            C      DISPLAY CIRCLE
000002             DIMENSION MESS(4)
000002             DATA MESS/40H TOO MANY FIGURES. CONSOLE RELEASED     /
000002             COMMON IBUF (100), MAD (2), IDDAD (10), NBYTE, MBYTE, NCON, K
000002             NBYTE = 0
000003             CALL GITCOF (NCON, ITH, ITV)
000006             CALL GURSET (ITH, ITV, 102B, IBUF, NBYTE, MBYTE)
000012             CALL GUMACG(MAD(2),1,IBUF,NBYTE,MBYTE)
000016             CALL GIDISP (NCON, IBUF, NBYTE, IDDAD (K+1), 2, 2, K+1)
000027             K = K+1
000031             IF (K.EQ.11) 1, 2
            C      TOO MANY FIGURES. RELEASE CONSOLE
000035           1 NBYTE=0
000036             CALL GURSET (0,-1400, 2,IBUF,NBYTE,MBYTE)
000042             CALL GUAN (MESS, 35,IBUF,NBYTE, MBYTE)
000046             CALL GIDISP(NCON,IBUF,NBYTE,IDDAD)
000051             CALL GICNRL (NCON)
000053             STOP
000055           2 RETURN
000056             END

                   SUBROUTINE LINE
            C      DISPLAY LINE
000002             COMMON IBUF (100), MAD (2), IDDAD (10), NBYTE, MBYTE, NCON, K
000002             DIMENSION MESS(4)
000002             DATA MESS/40H TOO MANY FIGURES. CONSOLE RELEASED     /
000002             NBYTE = 0
000003             CALL GITCOF (NCON, ITH, ITV)
000006             CALL GURSET (ITH, ITV, 102B, IBUF, NBYTE, MBYTE)
000012             CALL GUMACG(MAD(1),1,IBUF,NBYTE,MBYTE)
000016             CALL GIDISP (NCON, IBUF, NBYTE, IDDAD (K+1), 2, 1, K+1)
000027             K = K+1
000031             IF (K.EQ.11) 1, 2
            C      TOO MANY FIGURES. RELEASE CONSOLE
000035           1 NBYTE =0
000036             CALL GURSET (0,-1400, 2,IBUF,NBYTE,MBYTE)
000042             CALL GUAN (MESS,35, IBUF, NBYTE,MBYTE)
000046             CALL GIDISP(NCON,IBUF,NBYTE,IDDAD)
000051             CALL GICNRL (NCON)
000053             STOP
000055           2 RETURN
000056             END

                   SUBROUTINE ERASE
            C      CLEAR DISPLAY ITEM
000002             COMMON IBUF (100), MAD (2), IDDAD (10), NBYTE, MBYTE, NCON, K
            C      READ DISPLAY ITEM ID BLOCK PICKED
000002             CALL GIFID (NCON, IDDT, IDDC, IDWA)
000005             CALL GIERAS (IDDAD(IDWA)  )
000010             RETURN
000011             END

           7
           8
           9
           XMPL
           SCR
           6
           7
           8
           9
```

CIRCUIT APPLICATION DESCRIPTION

The CIRCUIT application permits a display console operator to design simple circuits consisting of nodes (connecting points), resistors, capacitors, and shorts. The application is intended for classroom use to guide students in the use of the 6000/1700 series IGS routines.

The application currently consists of five tasks. The fourth task can be modified to provide the capability to enter values and labels from a font for components in the circuit. The fifth task calls GIABRT for a voluntary abort.

Some conventions have been imposed to simplify the coding. The most important are:

- Circuits are always created from left to right and from top to bottom.
- If a component is to be added in a position already occupied by a component, the first component is deleted.
- A maximum of four components can be tied to any single node.
- The maximum number of nodes is limited to 25.
- If a node is erased, all components tied to that node are likewise erased.
- The first node should not be picked to be erased.

Table A-1 describes the tasks and their associated subroutines.

TABLE A-1. CIRCUIT TASK BREAKDOWN

Task	Main Program	Subroutines			
CRKT	CIRCUIT	NMAKE	SCHNDHV	ERRMESS	GETBEAD
MAKE	MAKE	CHECK GETBEAD	ERRMESS DELCOMP	SCHNDHV	NMAKE
ERAS	ERASE	DELCOMP	ERRMESS		

The application relies heavily on the data handler to store all the relational information required. Beads are formed for all nodes and components. Node beads are strung to implement a node search algorithm in the application. The bead address of the first node created is stored in COMMON as a state variable (ISTATEV) for the node bead string. Display addresses for all nodes and components are stored in the beads as well. (See Appendix B for a discussion of the data handler dump of this program.)

There is a COMMON for this application, but it is adequately presented in the listing and further discussion is unnecessary.

TASK CRKT

The initial task, CRKT, generates the byte-streams for all the components and enters them into the fixed address area of the display buffer as macros. It also displays nine light buttons and the first node.

The light buttons are:

HORS	Horizontal Resistor
VERS	Vertical Resistor
HOCP	Horizontal Capacitor
VECP	Vertical Capacitor
HOST	Horizontal Short
VEST	Vertical Short
ERAS	Erase
ENVL	Enter Value and Label
DONE	Voluntary Abort

There is no significance to the fact that all light buttons are four characters. They could have been spelled out in full.

CRKT also initializes the data handler for tasks MAKE and ERAS. Once complete, CKRT is never re-entered. Further, there is no access to CRKT except at sign-on time.

TASK MAKE

MAKE is the main program of the task. It determines which action the display console operator is requesting.

When the first six light buttons are displayed with GIDISP, the program name, MAKE, is stored in the IDWA parameter. If one of these light buttons is picked, the program MAKE is entered. MAKE calls AELBUT to return the IDDT and IDDC of the button. AELBUT renames IDDC as ICOD.

Assume the 274 operator picked the initial node (the only one displayed at this time) and the HORS (horizontal resistor) light button. The ID block associated with this button indicates an IDDT of 1 and an IDDC = ICOD = 1.

The nodes are displayed as single pick items; so to determine whether a node was picked, the user must call GIFID. To determine which component is to be attached to the node, the user calls GETBEAD where comparisons are made with component codes. If there is no match, ERRMESS displays the message NO NODE PICKED TRY AGAIN. This check prevents attaching a component to a component.

If there is a match, the user calls CHECK to be sure no component occupies the space where this component is to be displayed. CHECK also calls DMGET to retrieve the number of leads (IVAL) tied to the subject node. The user next determines whether a horizontal component or a vertical component is being considered and sets I so the proper lead will be added. If a horizontal component is requested, I = 7 to extract the right element bead address from the seventh word in the node bead. If the component is vertical, I = 8 to extract the down-element bead address from the eighth word in the node bead. (Refer to bead breakdowns in GETBEAD.)

When MAYBE contains a bead address, a component is to be deleted. For the case assumed above, MAYBE = 0. The number of beads tied to the node is also zero; therefore, IFLG is set to 1 to indicate that a terminal node must be created. IVAL is incremented by 1 to reflect this new bead. Return to MAKE.

MAKE calls GETBEAD with ICOD = 1 and GETBEAD renames ICOD as ICOMCOD. ICOMCOD is 1 in this case so a computed GO TO directs a call to DMGTBD. DMGTBD picks up a bead five words long. The code (1) for a horizontal resistor is stored in the first word. The "from node" bead address is stored in the second word as a pointer. IBDA is set to the component's bead address before the return to MAKE. Note that the display address of the resistor is not stored because it is not yet defined.

GURSET is called from MAKE to set the coordinates for the starting point of the component to be displayed. The strike coordinates on the node returned from GIFID are used. GURSET specifies a medium light intensity with ICODE = 2. The beads displayed are insensitive to light pen strikes. This is significant because when a node is picked, only the node – not the leads of the components tied to that node – should be pickable. To ensure strikes on nodes, the first and last 75 dgus of the beads on any component are not pickable.

GUMACG generates the macro calling sequence for the display of the element identified by MAD(ICOD). GIDISP then adds IDDT, IDDC, and IDWA to the component's byte stream. IDWA is used to store the component's bead address so the program can access node pointers and the component's display address.

IFLG is tested to see if a terminating node is required for the component. Since CHECK set IFLG to 1, a terminating node is required.

NMAKE creates the terminal node 400 dgus to the right of the starting node since this component is a horizontal resistor. NMAKE tests ISTATEV for a zero value. ISTATEV is zero at sign-on; thereafter it contains the bead address of the first node. In the assumed case, ISTATEV is non-zero so SCHNDHV is called to check for a node at (INH, INV).

The coordinates of the node are compared with those of beads in a string. If a match occurs, IBDB = ISTATEV and IMAKE = 1 which indicates that a node exists. If no match occurs, IMAKE = 0. Program control returns to NMAKE.

NMAKE checks for IMAKE = 0. If it is 0, GETBEAD gets a bead for the terminal node. DMSET then puts the next node pointer into the next-to-last bead created. The bead address of the last bead created is stored in ISTLSBD.

Then the macro for the terminal node is generated with the display address stored in word 4 of its bead and the number of beads set to 1. The node count (NDCNT) is increased by 1 and it is compared against the maximum number of nodes (25). TOO MANY NODES is displayed by ERRMESS if NDCNT = 25.

MAKE is entered and loads the rest of the pointers in the node and component beads.

AETSKR is called and the idle task is re-entered to wait for the next light button pick.

TASK ERAS

The main program of this task is ERAS. Task ERAS deletes the picked component or node and releases the associated bead or beads. Further, it removes pointers in other node and/or component beads for the deleted node and/or components. If the item to be deleted is a node, all components tied to it are deleted as well.

If the first node was picked, an error message is displayed. If any other node was picked, it is erased and its bead is released. The components attached to it are erased.

For example, ICODE = 3 (horizontal capacitor) and NOPE = 1. In subroutine DELCOMP, a call to DMGET displays the address of the capacitor. GIERAS erases the capacitor. KICK and ICOD were passed to DELCOMP through COMMON. The "into" node bead address is obtained from word 3 of the capacitor bead and the number of beads in the node is decreased by one. The decreased number of beads is stored, the left element pointer in the node bead is cleared, and the capacitor bead released to free storage. The number of beads in the "from" node are reduced and the pointer to the capacitor removed. Return to ERASE; reset NOPE to zero. Exit with a call to AETSKR.

```
*CONTROL CARDS

RUN(S)
LGO.
AEFILE
SCR.
RELEASE(CIRXXX)
EXIT.
DMP(177777)
RELEASE(CIRXXX)
7/8/9



COVERLAY (0,0)

      OVERLAY(SCR,0,0)
      PROGRAM CREATE
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
C
C          COMMON INFORMATION IS GIVEN IN TASK CIRCUIT (OVERLAY(1,0) )
C         BEAD INFORMATION IS GIVEN IN SUBROUTINE GETBEAD OF OVERLAY (0,0)
C
C         START BY TURNING CONTROL OVER TO GRAPHICS APPLICATION EXECUTIVE
      CALL MAIN
      END


CSUBROUTINE DELCOMP

      SUBROUTINE DELCOMP
C     DELETES COMPONENT FOR INSERTION OF ANOTHER IN ITS PLACE.
C     ITS BEAD AND POINTERS ALSO DELETED, DISPLAY DELETED.
C     LEAD COUNT IN PICKED NODE DECREASED TEMPORARILY FOR
C     CONSISTENCY IN ROUTINE CHECK WHICH INCREASES LEAD COUNT
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
C         DETERMINE IF COMPONENT IS HORIZONTAL OR VERTICAL
      I=7
      J=6
      IF((ICOD.AND.1).NE.0) GO TO 10
C         ICOD IS EVEN THEREFORE COMPONENT IS VERTICAL
      I=8
      J=5
C         IF NOPE=0 UNDESIREABLE ELEMENT ATTACHED TO PICKED NODE WILL
C         BE DELETED.  IF NOPE=1 PICKED ELEMENT WILL BE DELETED.
   10 IF(NOPE.EQ.0) CALL DMGET (ICOMP(I),IA,KICK)
      CALL DMGET(ICOMP(4),KICK,IBTSTRM)
C         ERASE ELEMENT
      CALL GIERAS(IBTSTRM)
```

```
C          UPDATE NODE BEAD STRING TO TAKE INTO ACCOUNT DELETED ELEMENT.
      CALL DMGET(ICOMP(3),KICK,POINTER)
      CALL DMGET(ICOMP(2),POINTER,ILEADS)
      CALL DMSET(ICOMP(2),POINTER,ILEADS-1)
      CALL DMSET (ICOMP(J),POINTER,0)
      IBDB = POINTER
      CALL DMGET(ICOMP(2),KICK,IA)
      CALL DMSET (ICOMP(I),IA,0)
      CALL DMRLBD(KICK)
C     SPLICE OUT UNNECESSARY TERMINAL NODE BEAD
      DO 20 I=5,8
      CALL DMGET (ICOMP(I),POINTER,IEL)
      IF(IEL.NE.0) GO TO 70
   20 CONTINUE
C          LOCATE PRECEDING AND FOLLOWING NODE BEADS.  NEXT NODE OF
C          PRECEDING BEAD IS CHANGED TO POINT TO FOLLOWING NODE
      NEXT = ISTATEV
   30 CALL DMGET (ICOMP(3),NEXT,NEXT1)
      IF(NEXT1.EQ.POINTER) GO TO 40
      NEXT=NEXT1
      GO TO 30
   40 IF(NEXT1.NE.ISTLSBD) GO TO 50
      ISTLSBD=NEXT
      CALL DMSET (ICOMP(3),NEXT,0)
      GO TO 60
   50 CALL DMGET(ICOMP(3),POINTER,IFXPTR)
      CALL DMSET(ICOMP(3),NEXT,IFXPTR)
   60 CALL DMGET (ICOMP(4),POINTER,IDISPAD)
C          TERMINAL NODE IS ERASED - BEAD IS RELEASED
      CALL GIERAS (IDISPAD)
      CALL DMRLBD(POINTER)
      NDCNT=NDCNT-1
      IF(NOPE.EQ.0) NOPE=1
   70 CALL DMGET(ICOMP(2),IA,ILEADS)
      CALL DMSET(ICOMP(2),IA,ILEADS-1)
      RETURN
      END


CROUTINE ERRMESS

      SUBROUTINE ERRMESS(IWHY)
C     THIS DISPLAYS THE CORRECT ERROR MESSAGE
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
      DIMENSION MESS(3,3)
      DATA (MESS(I),I=1,3)/30HTOO MANY NODES                /
     1,    (MESS(I),I=4,6)/30HNO NODE PICKED TRY AGAIN      /
     2,    (MESS(I),I=7,9)/30HWRONG PICK TRY AGAIN          /
C          DISPLAY ERROR MESSAGE
      NBYTE = 0
      CALL GURSET(-1000,1000,103B,IBUF,NBYTE,MBYTE)
      CALL GUAN (MESS(1,IWHY),24,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDISP,4,I)
```

```
C     MAKE MESSAGE A PRIME BUTTON
C     IF OPERATOR PICKS ERROR MESSAGE IT WILL BE ERASED
      CALL GIPBUT(NCON,4,1,10)
      CALL GIBUT(0,NCON,IDT,IDC)
      IF(IDC.EQ.10) CALL GIERAS(IDISP)
      CALL GIPBUT (NCON)
      RETURN
      END


CSUBROUTINE GETBEAD

      SUBROUTINE GETBEAD(ICOMCOD)
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,NBYTE,MBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
C     CREATES AND ENTERS PRELIMINARY INFO INTO BEADS WITH THIS FORMAT
C                 *NODE*                 *RES/CAP*            *SHORT*
C     WORD1       CODE=7                 CODE = C             CODE = CS
C     WORD2       NO.LEADS               FROM NODE(P)         FROM NODE(P)
C     WORD3       NEXTNODE               INTO NODE(P)         INTO NODE(P)
C     WORD4       IDDAD                  IDDAD                IDDAD
C     WORD5       UP ELEMENT(P)          LABEL/VALUE          NOT USED
C     WORD6       LEFT ELEMENT(P)        VALUE/IDDAD          NOT USED
C     WORD7       RIGHT ELEMENT(P)
C     WORD8       DOWN ELEMENT(P)
C     WORD9       IHCEN                NOTE- POINTERS POINT TO IBEAD WORDS
C     WORD10      IVCEN
C
C                 P = POINTER            C = 1 HORES          CS = 5 HORS
C                                        C = 2 VERES          CS = 6 VERS
C                                        C = 3 HORCAP
C                                        C = 4 VERCAP
C
      IF(ICOMCOD.GT.6) GO TO 30
      IF(ICOMCOD.GT.4) GO TO 10
C         GET RESISTOR OR CAPACITOR BEAD
      CALL DMGTBD(5,IBD)
      GO TO 20
C         GET SHORT BEAD
   10 CALL DMGTBD(6,IBD)
   20 CALL DMSET(ICOMP(1),IBD,ICOMCOD)
      CALL DMSET(ICOMP(2),IBD,IA)
      IBDA = IBD
      RETURN
C         GET NODE BEAD
   30 CALL DMGTBU(10,IBD)
      CALL DMSET(ICOMP(1),IBD,ICOMCOD)
      CALL DMSET(ICOMP(9),IBD,INH)
      CALL DMSET(ICOMP(10),IBD,INV)
      CALL DMSET(ICOMP(2),IBD,1)
      IBDB = IBD
      RETURN
      END
```

```
CSUBROUTINE NMAKE

      SUBROUTINE NMAKE(INHL,INVL)
C     USED TO CHECK FOR NODE AT INH,INV. IF NODE THERE, A BEAD IS
C     FETCHED FOR IT AND NODE IS DISPLAYED. NODES ARE STRUNG.
C     STATE VARIABLE (FIRST NODE BEAD ADDRESS) IS STORED IN COMMON
C     LOCATION ISTATEV
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
      NDMAX = 25
      INH = INHL
      INV = INVL
      IF(ISTATEV.NE.0) GO TO 20
C         LOGIC FOR FIRST NODE
C         GET BEAD, FIRST AND LAST BEAD ADDRESSES SET TO BEAD ADDRESS OF
C         THIS NODE
      CALL DMGTBD(10,IBD)
      ISTATEV = IBD
      ISTLSBD = IBD
C         SET CODE AND COORDINATES OF NODE
      CALL DMSET(ICOMP(1),IBD,7)
      CALL DMSET(ICOMP(9),IBD,INH)
      CALL DMSET(ICOMP(10),IBD,INV)
C         DISPLAY NODE
   10 NBYTE=0
      INHH = INHH2
      CALL GURSET( INHH,INV,102B,IBUF,NBYTE,MBYTE)
      CALL GUMACG(MAD(7),1,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,NIDDAD,2,7,IBD)
C         SET DISPLAY ADDRESS
      CALL DMSET(ICOMP(4),IBD,NIDDAD)
      NDCNT = NDCNT + 1
C         CHECK TO SEE IF MAXIMUM NUMBER OF NODES HAS BEEN EXCEEDED
      IF (NDCNT .EQ. NDMAX) CALL ERRMESS (1)
      GO TO 30
C         CHECKS FOR EXISTENCE OF TERMINAL NODE OF ELEMENT
   20 CALL SCHNDHV
C         IF THERE IS ONE-RETURN
      IF (IMAKE.NE.0) GO TO 25
C     MAKE A BEAD FOR NODE
      CALL GETBEAD(7)
      CALL DMSET(ICOMP(3),ISTLSBD,IBDB)
      ISTLSBD = IBDB
      GO TO 10
C     WHEN NODE IS ALREADY PRESENT, ADD ONE TO NUMBER OF BEADS
   25 CALL DMGET(ICOMP(2),IBDB,IVAL)
      CALL DMSET(ICOMP(2),IBDB,IVAL +1)
   30 RETURN
      END

CSUBROUTINE SCHNDHV

      SUBROUTINE SCHNDHV
C     CHECKS FOR EXISTENCE OF NODE AT INH,INV
C     FLAG IMAKE EQUALS ZERO, TO MAKE BEAD, EQUALS ONE FOR DONT MAKE
```

```
C     BEAD. IBDB HOLDS EXISTING BEAD ADDRESS IF IMAKE EQUALS ONE
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
C         LOOP THROUGH NODE BEADS TO DETERMINE IF A NODE EXISTS AT END OF
C           ELEMENT
      NEXT = ISTATEV
   10 CALL DMGET(ICOMP(9),NEXT,IH)
      CALL DMGET(ICOMP(10),NEXT,IV)
      IF((INH.GE.IH-60 .AND. INH .LE. IH+ 60) .AND.
     1    (INV .GE. IV-60 .AND. INV .LE.IV+60) ) GO TO 20
      CALL DMGET(ICOMP(3),NEXT,NEXT)
      IF (NEXT.NE.0) GO TO 10
C         NO NODE EXISTS-IMAKE=0
      IMAKE = 0
      RETURN
C       A NODE EXISTS IMAKE=1 AND SAVE BEAD ADDRESS OF NODE
   20 IBDB = NEXT
      IMAKE = 1
      RETURN
      END


COVERLAY (1,0)


      OVERLAY(1,0)
      PROGRAM CIRCUIT
C     PERMITS USER TO DESIGN SIMPLE CIRCUITS CONSISTING OF NODES (TIE
C     POINTS), RESISTORS, CAPACITORS, AND SHORTS, USER MUST START AT
C     DISPLAYED NODE AND COMPONENTS MUST BE ADDED FROM LEFT TO RIGHT OR
C     TOP TO BOTTOM. IF USER INDICATES DESIRE TO ADD COMPONENT TO
C     POSITION OCCUPIED BY ANOTHER COMPONENT, THE FIRST COMPONENT IS
C     DELETED
C         ERROR MESSAGES ARE SUPPLIED IN CASE OPERATOR GOOFS.  A MESSAGE
C         MAY BE ERASED BY PICKING IT BEFORE ANY BUTTON IS PICKED.
C
C

CCOMMON VARIABLES
C
C
C     IBUF - - - TEMPORARY BYTE BUFFER
C     MAD - - - MACRO AND SUBROUTINE ADDRESS
C                (1)      HORIZONTAL RESISTOR
C                (2)      VERTICAL RESISTOR
C                (3)      HORIZONTAL CAPACITOR
C                (4)      VERTICAL CAPACITOR
C                (5)      HORIZONTAL SHORT
C                (6)      VERTICAL SHORT
C                (7)      NODE
C     IA - - - - BEAD ADDRESS OF PICKED NODE
C     INH - - -  HORIZONTAL COORDINATE OF NODE TO BE DISPLAYED
C     INV - - -  VERTICAL COORDINATE OF NODE TO BE DISPLAYED
C     IBDA - - - BEAD ADDRESS OF CURRENT COMPONENT
C     IBDB - - - BEAD ADDRESS OF TERMINAL NODE FOR COMPONENT
C     ISTATEV -  STATE VARIABLE,CONTAINS BEAD ADDRESS OF FIRST NODE IN
C                  STRING OF NODES
```

```
C      ISTLSBD -  BEAD ADDRESS OF LAST NODE IN STRING OF NODES
C      IMAKE - -  FLAG TO INDICATE TO MAKE BEAD OR NOT
C            = 0  MAKE BEAD
C            = 1  DO NOT MAKE BEAD, ONE ALREADY EXISTS AT INPUT
C                 COORDINATES
C      POINTER -  CONTAINS BEAD ADDRESS OF COMPONENTS TO BE DELETED
C                 BECAUSE ASSOCIATED NODE IS DELETED
C      IFLG - - - FLAG TO INDICATE WHETHER TERMINAL NODE EXISTS FOR
C                 COMPONENT
C             = 0 TERMINAL NODE EXITS
C             = 1 TERMINAL NODE DOES NOT EXIST
C      NCON - - - CONSOLE NUMBER
C      NDCNT - -  NO. OF NODES
C      NDMAX - -  MAXIMUM NUMBER OF NODES
C      MBYTE - -  MAXIMUM BYTES ALLOWED IN IBUF
C      NBYTE - -  CURRENT NUMBER OF BYTES IN IBUF
C      ICOD - - - COMPONENT CODE OF BUTTON PICKED
C      IBD - - -  BEAD ADDRESS OF BEAD CURRENTLY BEING CREATED
C      NOPE - - - SET BY ERASE TO SKIP STEPS NOT NEEDED IN DELCOMP
C      KICK - - - IA PARAMETER PASSED TO DELCOMP
C
C
C
CEND OF COMMON VARIABLES
C
C
C
       COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
      1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
      2NOPE,KICK,ICOMP(12)
       DIMENSION  NEWM(3),NAME(60)
       DATA (NAME(I),I=1,9) /4HHORS,4HVERS,4HHOCP,4HVECP,4HHOST,4HVEST,
      1 4HERAS,4HENVL,4HDONE/
       DATA  (NEWM(I),I=1,3)/4RERAS,4RENVL,4RDONE/
C          SET UP CONSOLE BY NCON CONSOLE CARD WANTED GOES LAST
       NCON=1
       NCON=2
       CALL GICNJB(NCON)
C
C      SET INPUT CONTROL MASKS
C
       CALL GIMASK (NCON,-0,2,2)
       CALL GIMASK (NCON,-0,4,4)
       CALL GIMASK (NCON,-0,1,8)
       CALL GIMASK (NCON,-0,7,16)
       ICOMP2 =010000000000B
       DO 10  I=1,12
       ICOMP(I) = I .OR. ICOMP2
    10 CONTINUE
       K = 0
       ISTATEV = 0
       MBYTE = 310
       NDCNT = 0
       NOPE = 0
       KICK = 0
       ISEE = 4302B
       IDNTSEE = 4202B
       DO  20  I = 1,60
    20 MAD (I) = 0
```

```
C     CREATE HOR. RES. BYTESTREAM WITH LEADS UNPICKABLE
      NBYTE=0
      CALL GUSEGS(0,0,75,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(75,0,0)
      CALL GUBYTE( ISEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(75,0,-0,IBUF,NBYTE,MBYTE)
      N = -50
      DO 30 I = 1,9
      N = -N
   30 CALL GUSEG (75+25*I,N,1)
      CALL GUSEG(325,0,1)
      CALL GUSEG(325,0,0)
      CALL GUBYTE(IDNTSEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(325,0,400,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(1))
C
C     CREATE HOR. CAP BYTESTREAM WITH LEADS UNPICKABLE
C
      NBYTE=0
      CALL GUSEGS(0,0,75,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(75,0,0)
      CALL GUBYTE( ISEE,I,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(75,0,175,0,1,-0,IBUF,BNYTE,MBYTE)
      CALL GUSEG(175,50,0)
      CALL GUSEG(175,-50,1)
      CALL GUSEG(225,-50,0)
      CALL GUSEG(225,50,1)
      CALL GUSEG(225,0,0)
      CALL GUSEG(325,0,1)
      CALL GUSEG(325,0,0)
      CALL GUBYTE(IDNTSEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(325,0,400,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(3))
C
C     MAKE NODE BYTESTREAM
      NBYTE=0
      CALL GUARCG(1,0,0,12,0,12,0,-0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(7))
C
C     MAKE VER. RES. WITH LEADS UNPICKABLE
C
      NBYTE = 0
      CALL GUSEGS ( 0,400,0,325,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(0,325,0)
      CALL GUBYTE( ISEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(0,325,-0,IBUF,NBYTE,MBYTE)
      N =-50
      DO 40 I = 1,9
      N = -N
   40 CALL GUSEG (N,325-25*I,1)
      CALL GUSEG ( 0,75,1)
      CALL GUSEG(0,75,0)
      CALL GUBYTE(IDNTSEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(0,75,0,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(2))
C
C     CREATE VER. CAP. WITH UNPICKABLE LEADS
```

```
C
      NBYTE=0
      CALL GUSEGS(0,0,0,-75,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(0,-75,0)
      CALL GUBYTE( ISEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(0,-75,0,-175,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(50,-175,0)
      CALL GUSEG(-50,-175,1)
      CALL GUSEG(-50,-225,0)
      CALL GUSEG(50,-225,1)
      CALL GUSEG(0,-225,0)
      CALL GUSEG(0,-325,1)
      CALL GUSEG(0,-325,0)
      CALL GUBYTE(IDNTSEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(0,-325,0,-400,1,-0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(4))
C
C     MAKE HORIZONTAL SHORT WITH ENDS UNPICKABLE
C
      NBYTE=0
      CALL GUSEGS(0,0,75,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(75,0,0)
      CALL GUBYTE( ISEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(75,0,325,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(325,0,0)
      CALL GUBYTE(IDNTSEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(325,0,400,0,1,-0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(5))
C
C     MAKE VERTICAL SHORT WITH ENDS UNPICKABLE
C
      NBYTE=0
      CALL GUSEGS(0,0,0,-75,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(0,-75,0)
      CALL GUBYTE( ISEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(0,-75,0,-325,1,-0,IBUF,NBYTE,MBYTE)
      CALL GUSEG(0,-325,0)
      CALL GUBYTE(IDNTSEE,1,IBUF,NBYTE,MBYTE)
      CALL GUSEGS(0,-325,0,-400,1,0,IBUF,NBYTE,MBYTE)
      CALL GIMAC (NCON,IBUF,NBYTE,MAD(6))
C
C     INITIALIZE DATA MANAGER
      CALL DMINIT(4HSTAN,2)
C
C
C     CREATE AND DISPLAY 9 LIGHTBUTTONS
C
      IHL=1200
      DO 50 I = 1,6
      NBYTE=0
      CALL GURSET ( -1500,IHL,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN ( NAME(I),4,IBUF,NBYTE,MBYTE)
      CALL GIDISP ( NCON,IBUF,NBYTE,IDDAD,1,I,4RMAKE)
   50 IHL=IHL-100
      IHL=600
      DO 60 I = 7,9
      NBYTE=0
      CALL GURSET ( -1500,IHL,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN (NAME(I),4,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,1,I,NEWM(I-6))
   60 IHL=IHL-100
      NBYTE = 0
      IHL = IHL - 100
      CALL GURSET(-1500,IHL,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(5HDMDMA,5,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,1,77,4RDUMP)
C     DISPLAY FIRST MODE AT -800,800
      CALL NMAKE (-800,800)
C
C     DISPLAY FIRST NODE AT -800,800
C
      CALL NMAKE(-800,800)
C
C     USER HAS NOW PICKED NODE AND INDICATED WHAT KIND OF
C     COMPONENT HE WANTS ATTACHED TO IT BY SELECTING THE
C     PROPER LIGHTBUTTON.
      CALL AETSKR
      END


COVERLAY(2,0)

      OVERLAY(2,0)
      PROGRAM MAKE
C     MAKES BYTESTREAM AND BEAD FOR INDICATED ELEMENT
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
C         DETERMINE WHAT BUTTON WAS PICKED
      CALL AELBUT(IDDT,ICOD)
C         DETERMINE IF A NODE WAS PICKED
      CALL GIFID(NCON,INDTP,ICD,IA,IB,IPH,IPV)
      IF(INDTP.EQ.2.AND.ICD.EQ.7) GO TO 10
      CALL ERRMESS(2)
      GO TO 50
C         CHECK TO SEE IF UNDESIRABLE ELEMENT IS CONNECTED TO NODE
C         IF SO, DELETE
   10 CALL CHECK
C     GET CENTER COORDINATES OF NODE
      CALL DMGET(ICOMP(9),IA,IPH)
      CALL DMGET(ICOMP(10),IA,IPV)
C         OBTAIN AN ELEMENT BEAD
      CALL GETBEAD(ICOD)
          DISPLAY ELEMENT AND SAVE DISPLAY ADDRESS
      NBYTE = 0
      IF(ICOD.AND.1).EQ.0) GO TO 15
      IPHH = IPHH2
      IPVV = IPV
      GO TO 17
   15 IPHH = IPH
      IPVV = IPV - 12
   17 CONTINUE
      CALL GURSET(IPH,IPV,2,IBUF,NBYTE,MBYTE)
```

```
C          OBTAIN AN ELEMENT BEAD
      CALL GETBEAD(ICOD)
C          DISPLAY ELEMENT AND SAVE DISPLAY ADDRESS
      NBYTE = 0
      CALL GURSET(IPH,IPV,2,IBUF,NBYTE,MBYTE)
      CALL GUMACG(MAD(ICOD),1,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,NIDDAD,2,ICOD,IBDA)
      CALL DMSET(ICOMP(4),IBDA,NIDDAD)
C MAKE NODE AT END OF ELEMENT IF NEEDED
      IF(IFLG.EQ.0) GO TO 30
      IF((ICOD.AND.1).EQ.0) GO TO 20
      CALL NMAKE((IPH+424),IPV)
      GO TO 30
   20 CALL NMAKE (IPH,(IPV-424))
C FILL IN POINTERS ON NODES AND ELEMENT
   30 CALL DMSET(ICOMP(1),IBDA,ICOD)
      CALL DMSET(ICOMP(2),IBDA,IA)
      CALL DMSET(ICOMP(3),IBDA,IBDB)
      IF((ICOD.AND.1).EQ.0) GO TO 40
C          SAVE NECESSARY ITEMS OF HORIZONTAL COMPONENT
      CALL DMSET(ICOMP(7),IA,IBDA)
      CALL DMSET(ICOMP(6),IBDB,IBDA)
      GO TO 50
C          SAVE NECESSARY ITEMS OF VERTICAL COMPONENT.
   40 CALL DMSET(ICOMP(8),IA,IBDA)
      CALL DMSET(ICOMP(5),IBDB,IBDA)
   50 CALL AETSKR
      END
```

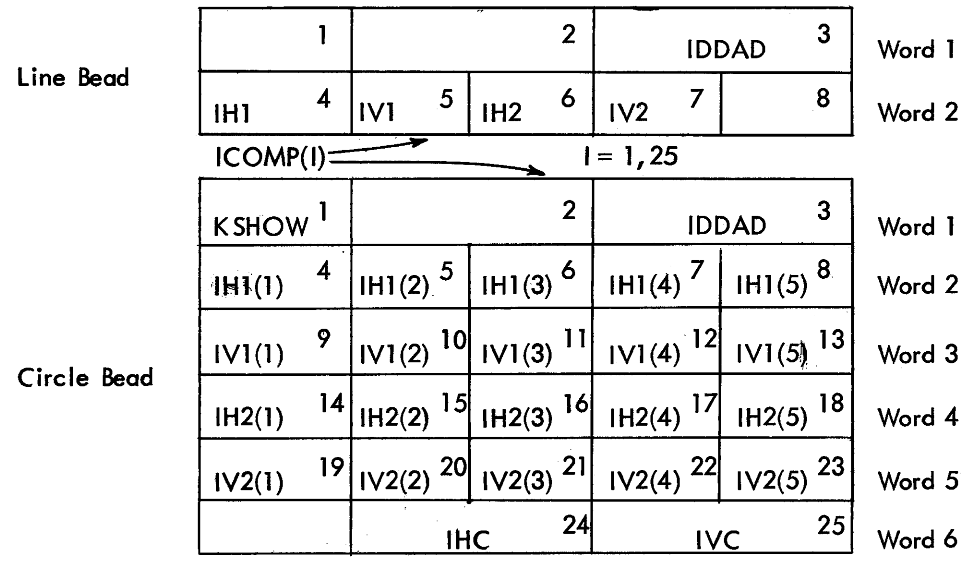
```
CSUBROUTINE CHECK

      SUBROUTINE CHECK
C  CHECKS FOR EXISTENCE OF A COMPONENT POINTER IN A NODE BEAD
C AND CHECKS FOR NUMBER OF COMPONENTS TIED TO A NODE. IN EITHER
C CASE, UNDESIRABLE COMPONENT IS DELETED.
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(70),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
      CALL DMGET(ICOMP(2),IA,IVAL)
C          IF 4 LEADS ARE CONNECTED TO NODE MUST DELETE UNDESIRABLE
C          COMPONENT
      IF(IVAL.NE.4) GO TO 20
   10 CALL DELCOMP
C SET IFLG TO ZERO FOR NO TERMINATE NODE NEEDED
C     WHEN ELEMENT ADDED
      IFLG = 0
      IF(NOPE.EQ.0) GO TO 40
      NOPE=0
      GO TO 30
C          DETERMINE IF HORIZONTAL OR VERTICAL COMPONENT IS BEING CONSIDER
   20 I=7
      IF((ICOD.AND.1).EQ.0) I=8
      CALL DMGET (ICOMP(I),IA,MAYBE)
C          IF AN UNDESIRABLE ELEMENT IS CONNECTED TO NODE DELETE IT.
      IF(MAYBE.NE.0) GO TO 10
C SET IFLG TO ONE TO MAKE TERMINATE NODE
```

```
C          WHEN ELEMENT IS ADDED.
   30 IFLG = 1
   40 CALL DMGET(ICOMP(2),IA,IVAL)
      IVAL = IVAL + 1
      CALL DMSET(ICOMP(2),IA,IVAL)
      RETURN
      END

COVERLAY (3,0)

      OVERLAY(3,0)
      PROGRAM ERAS
C     ERASE PICKED COMPONENT OR NODE AND RELEASES BEAD
C     IF A NODE, ALL COMPONENTS ATTACHED TO IT ARE ERASED
      INTEGER POINTER
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
      CALL GIFID(NCON,INDTP,IDC,IA,IB)
      IF(.NOT.(INDTP .EQ. 2 .AND. IDC .EQ. 7)) GO TO 60
C          IF FIRST NODE IS PICKED PRINT ERROR
      IF(IA.EQ.ISTATEV) GO TO 70
C          A NODE WAS PICKED
C          RELEASE BEADS OF ELEMENTS ATTACHED TO NODE-ERASE ELEMENTS
      J=2
      DO 10 I=1,4
      CALL DMGET (ICOMP(I+4),IA,POINTER(I))
      IF(POINTER(I).EQ.0) GO TO 10
      CALL DMGET (ICOMP(4),POINTER(I),IDISPAD)
      CALL DMGET(ICOMP(1),POINTER(I),KIND)
      IF (KIND.GT.4)GO TO 5
      CALL DMGET(ICOMP(6),POINTER(I),IDAD)
      IF(IDAD.NE.0)CALL GIERAS(IDAD)
    5 CONTINUE
      IF(I.EQ.3)  J=3
      CALL DMGET (ICOMP(J),POINTER(I),IWHERE)
      CALL DMSET(ICOMP(9-I),IWHERE,0)
      CALL DMGET (ICOMP(2),IWHERE,ILEADS)
      CALL DMSET (ICOMP(2),IWHERE,ILEADS-1)
      CALL GIERAS (IDISPAD)
      CALL DMRLBD (POINTER(I))
   10 CONTINUE
C          LOCATE PRECEDING AND FOLLOWING NODE BEADS. NEXT NODE OF
C          PRECEDING NODE IS CHANGED TO POINT TO FOLLOWING NODE
      NEXT = ISTATEV
   20 CALL DMGET (ICOMP(3),NEXT,NEXT1)
      IF(NEXT1.EQ.IA) GO TO 30
      NEXT=NEXT1
      GO TO 20
   30 IF(IA.NE.ISTLSBD) GO TO 40
      ISTLSBD=NEXT
      CALL DMSET (ICOMP(3),NEXT,0)
      GO TO 50
   40 CALL DMGET(ICOMP(3),IA,IFIXPTR)
      CALL DMSET(ICOMP(3),NEXT,IFIXPTR)
   50 CALL DMGET(ICOMP(4),IA,IDISPAD)
C          NODE POINTED TO IS ERASED, BEAD IS RELEASED.
      CALL GIERAS(IDISPAD)
      CALL DMRLBD(IA)
      NDCNT=NDCNT-1
      GO TO 90
C         AN ELEMENT WAS PICKED
   60 IF(INDTP.EQ.2) GO TO 80
   70 CALL ERRMESS(3)
      GO TO 90
C         SET UP NECESSARY VARIABLES TO DELETE ELEMENT - THEN DELETE
   80 ICOD = IDC
      NOPE = 1
      KICK = IA
      CALL DELCOMP
      NOPE = 0
   90 CALL AETSKR
      END


COVERLAY(4,0)


      OVERLAY(4,0)
      PROGRAM ENVL
      COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
     1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
     2NOPE,KICK,ICOMP(12)
      CALL GIFID (NCON,IDDT,IDDC,IA)
      IF(IDDT.EQ.2.AND.IDDC.LT.5) GO TO 100
      CALL ERRMESS(3)
      GO TO 1000
  100 CALL GFONTN (NCON,0,-1300,IFTN)
      NX=10
      CALL GIANS (NCON,NX,800,-1000)
      NBYTE=0
      CALL GURSET (-100,-1600,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN (6HACCEPT,6,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,IACP,1,10)
      NBYTE=0
      CALL GURSET (100,-1600,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN (6HREJECT,6,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,IRJT,1,11)
  200 CALL GIBUT (0,NCON,ICO,IWD)
      IF(IWD.EQ.10.OR.IWD.EQ.11) GO TO 300
      CALL ERRMESS(3)
      GO TO 200
  300 CALL GIANE (NCON,NX,JBUF)
      ITYP = 1
      CALL EICHEK ( ITYP,JBUF,NX,IBUF)
      IF ( ITYP) 305,305,310
  305 CALL ERRMESS(3)
      CALL GIANS (NCON,NX,800,-1000)
      GO TO 200
  310 CALL GIERAS (IFTN,IACP,IRJT)
      IF(IWD.EQ.11) GO TO 100
      CALL DMGET (ICOMP(2),IA,NN)
      CALL DMGET (ICOMP(9),NN,JH)
      CALL DMGET (ICOMP(10),NN,JV)
      JH = JH + 75
      JV = JV + 75
      IF(IDDC.EQ.2.OR.IDDC.EQ.4) JV=JV-275
      CALL DMGET ( ICOMP(6),IA,IDV)
      CALL GIERAS ( IDV)
      NBYTE=0
      CALL GURSET (JH,JV,002B,IBUF,NBYTE,MBYTE)
      CALL GUAN (JBUF,NX,IBUF,NBYTE,MBYTE)
      CALL GIDISP (NCON,IBUF,NBYTE,IDV)
      CALL DMSET (ICOMP(6),IA,IDV)
  333 FORMAT(24(E,I2,3H,1))
      ENCODE(7,333,XX)NX
      DECODE(NX,XX,JBUF)VAL
      CALL DMSET (ICOMP(5),IA,VAL)
 1000 CONTINUE
      CALL AETSKR
      END


      SUBROUTINE ETCHEK(IOPE,BCO,NC,ITEM)
C
C THIS ROUTINE TRIES TO PROTECT A USER FROM FATAL ERRORS DUE TO AN
C ILLEGAL NUMBER BEING RECEIVED FROM A LIGHT REGISTER.  THIS ROUTINE
C CHECKS FOR SOME COMMON ERRORS THAT OCCUR WITH BOTH E FORMAT AND
C I FORMAT.
C
C       IORE    --IORE=0 MEANS CHECK AGAINST I FORMAT
C                 IORE.GT.0 MEANS CHECK AGAINST E FORMAT
C       BCD     --DISPLAY CODE OF ENTERED NUMBER. DIMENSIONED TO
C                 ALLOW MORE THAN TEN CHARACTERS.
C       NC      --NUMBER OF USABLE CHARACTERS IN BCD. MUST BE POSITIVE
C       ITEM    --TEMPORARY WORKING AREA. NEED AT LEAST NC WORDS.
C *****   NOTE   *****
C IORE IS RETURNED AS A NEGATIVE VALUE(-1) WHEN THE NUMBER IS BAD.
C IORE IS UNCHANGED FOR ACCEPTABLE NUMBERS.
C
      DIMENSION BCD(1),ITEM(1)
      NP=1
      L=0
      DO 100 I=1,NC
      L=L+1
      NS=61-6*L
      ITEM(I)=LBYT(NS,6,BCD(NP))
      IF (L-10)   100,50,50
   50 L=0
      NP=NP+1
  100 CONTINUE
C WE HAVE GENERATED NC WORDS IN ITEM. EACH CONTAINS ONE CHARACTER OF
C DIGIT FROM BCD.
      NP=0
      NS=0
      DO 1000 I=1,NC
      IT=ITEM(i)
      IF(IT.NE.45B.AND.IT.NE.46B)    GO TO 500
      NS=NS+1
      GO TO 1000
  500 IF (IT.EQ.57B)   NP=NP+1
 1000 CONTINUE
      IF (IORE)  7000,1500,2000
C CHECK INTEGER VALUE FOR TOO MANY DECIMAL POINTS OR SIGNS.
 1500 IF (NP)  7000,1500,2000
 1600 IF (NS-1)   8000,8000,7000
```

```
C CHECK FLOATING VALUE FOR TOO MANY DECIMAL POINTS OR SIGNS.
 2000 IF (NP-1)    2500,2500,7000
 2500 IF (NS-2)   8000,8000,7000
 7000 IORE=-1
 8000 RETURN
      END


COVERLAY(5,0)

    OVERLAY(5,0)
    PROGRAM DONE
    COMMON NCON,IBUF(100),MAD(60),IBDA,IBDB,ISTATEV,ISTLSBD,IMAKE,POIN
   1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,IA,INV,INH,
   2NOPE,KICK,ICOMP(12)
      VOLUNTARY ABORT OF JOB
    NBYTE=0
    CALL GURSET(-500,-1400,18,IBUF,NBYTE,MBYTE)
    CALL GUAN(3IN THATS ALL FOLKS,31,
   1IBUF,NBYTE,MBYTE)
    CALL GIDISP(NCON,IBUF,NBYTE,IDDAD)
    CALL GICNRL(NCON)
    STOP
    END

    OVERLAY (6,0)
    PROGRAM DUMP
    COMMON NCON,IBUF(100),MAD(60),IBDA,IBOB,ISTATEV,ISTLSB
   1TER(4),IFLG,NDCNT,NDMAX,MBYTE,NBYTE,ICOD,K,IBD,IPV,IPH,
   2NOPE,KICK,ICOMP(12)
C
    CALL DMFLSH
    CALL DMINIT (4HSTAN,2)
C
    CALL DMDMP
C
    CALL AETSKR
    END
7
 8
  9
CIRXXX
SCR
7
 8
  9
CIRXXX
6
 7
  8
   9
```

OVERLAY AND SUBROUTINE USE

The following program listing is designed to illustrate the use to which overlays and subroutines may be put in the Interactive Graphics system. Refer to Appendix B and the IGS Reference Manual for information concerning the data handler routines and beads. Check overlays (10, 10) and (11, 10) for special uses of ENCODE and DECODE (Appendix C contains further applications of ENCODE and DECODE). Look for the programming error in OVERLAY (11, 0)'s program ROTA. Software zooming and rotation are illustrated in this program. The following diagram illustrates the beads constructed by the data handler.



```
        JOB,P17,T10000,CM40000,TP1.
        RUN(S)
        LGO.
        AEFILE.
        SCR.
        RELEASE(KCB)
        EXIT.
        RELEASE(KCB)
7
 8
  9


       OVERLAY(SCR,0,0)
       PROGRAM CREATE (INPUT,OUTPUT)
       COMMON IBUF(64),NCOM,ICOMP(25),ID(80),NLIN,NCIR
       CALL AEXEC
       END


       OVERLAY(1,0)
       PROGRAM FIRST
       COMMON IBUF(64),NCOM,ICOMP(25),ID(80),NLIN,NCIR
       NCON=9
       CALL AETSKC(4LINIT)
       END
```

```
      OVERLAY(2,0)
      PROGRAM INIT
C * * THIS TASK INITIALIZES DISPLAY ON CRT - OPERATOR MAY RE-INITIALIZE
C * * IF HE DOES NOT LIKE CURRENT DISPLAY
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION NAME(9),IDWA(9)
      DATA (NAME(I),I=1,9)/4HLINE,4HCIRC,4HERAS,4HMOVE,4HCOPY,
     *4HZOOM,4HROTA,4HINIT,4HQUIT/
      DATA(IDWA(I),I=1,9)/4RLINE,4RCIRC,4RERAS,4RMOVE,4RCOPY,
     *4RZOOM,4RROTA,4RINIT,4RQUIT/
C * * ATTACH CONSOLE
      CALL GICNJB(NCON)
C * * SET UP MASKS FOR DETERMINING HOW TO TREAT PICKED ITEMS
      DO 10 I=1,5
      N=2**(I-1)
   10 CALL GIMASK(NCON,-0,N,N)
      DO 12 I=1,80
   12 ID(I)=0
C * * DISPLAY BUTTON MENU
      IH=-1500
      IV=+800
      DO 15 I=1,9
      NBYTE=0
      CALL GLRSET(IH,IV,102B,IBUF,NBYTE,312)
      CALL GLAN(NAME(I),4,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,30B,I,IDWA(I))
   15 IV=IV-100
C * * INITIALIZE FILE USED BY DATA MANAGER , CENTER TRACKING CROSS
      CALL DMINIT(4LDATA,2)
      CALL GITCON(NCON,0,0)
      NLIN=0
      NCIR=0
C * * SET UP COMPONENT CODES USED BY BEADS IN DATA MANAGER
      ICMP(1)=066014000001B
      ICMP(2)=053030000001B
      ICMP(3)=050030000001B
      ICMP(4)=066014000002B
      ICMP(5)=064414000002B
      ICMP(6)=063014000002B
      ICMP(7)=061414000002B
      ICMP(8)=060014000002B
      ICMP(9)=066014000003B
      ICMP(10)=064414000003B
      ICMP(11)=063014000003B
      ICMP(12)=061414000003B
      ICMP(13)=060014000003B
      ICMP(14)=066014000004B
      ICMP(15)=064414000004B
      ICMP(16)=063014000004B
      ICMP(17)=061414000004B
      ICMP(18)=060014000004B
      ICMP(19)=066014000005B
      ICMP(20)=064414000005B
      ICMP(21)=063014000005B
      ICMP(22)=0614140000005B
      ICMP(23)=0600140000005B
      ICMP(24)=0630300000006B
      ICMP(25)=0600300000006B
      CALL AETSKR
      END


      OVERLAY(3,0)
      PROGRAM LINE
C * * THIS TASK OVERLAY DISPLAYS A LINE ON THE CRT
      COMMON IBUF(64),NCOM,ICMP(25),ID(80),NLIN,NCIR
C * * RETRIEVE TRACKING CROSS COORDINATES OF INITIAL POINT OF LINE SEGMENT
C * * AND MOVE TRACKING CROSS BACK TO CENTER OF CRT
      CALL GITCOF(NCOM,IH1,IV1)
      CALL GITCON(NCOM,0,0)
C * * DISPLAY ENDL AT BEGINNING OF LINE SEGMENT
      NBYTE=0
      CALL GLRSET(IH1-50,IV1,103B,IBUF,NBYTE,312)
      CALL GLAN(4HENDL,4,IBUF,NBYTE,312)
      CALL GIDISP(NCOM,IBUF,NBYTE,IDPBUT,30B,10)
C * * PROGRAM RESTRICTS NUMBER OF LINES DISPLAYED TO MAXIMUM OF 40
      NLIN=NLIN+1
      IF(NLIN.GE.0.AND.NLIN.LE.40) GO TO 10
      PRINT 97
    8 CALL AETSKC(4LQUIT)
C * * WAIT FOR ENDL BUTTON TO BE PICKED, THEN ERASE IT FROM TUBE
   10 CALL GIBUT(0,NCOM,IDT,IDC)
      IF(IDT.NE.30B.OR.IDC.NE.10)GO TO 10
      CALL GIERAS(IDPBUT)
C * * RETRIEVE TRACKING CROSS COORDINATES OF END POINT OF LINE SEGMENT
      CALL GITCOF(NCOM,IH2,IV2)
C * * GET LINE BEAD AND STORE BEAD ADDRESS IN ID ARRAY
      CALL DMGTBD(2,IBD)
      DO 12 IDN=1,80
      IF (ID(IDN).EQ.0) GO TO 15
12    CONTINUE
      PRINT 98
      GO TO 8
   15 ID(IDN)=IBD
C * * DISPLAY SEGMENT ON CRT
      NBYTE=0
      CALL GLRSET(IH1,IV1,102B,IBUF,NBYTE,312)
      CALL GLSEGS(IH1,IV1,IH2,IV2,1,-0,IBUF,NBYTE,312)
      CALL GIDISP(NCOM,IBUF,NBYTE,IDDAD,22B,1,IDN)
C * * FILL LINE BEAD WITH DESCRIPTION OF SEGMENT, AND DISPLAY ADDRESS
      CALL DMSET(ICMP(1),IBD,2)
      CALL DMSET(ICMP(3),IBD,IDDAD)
      CALL DMSET(ICMP(4),IBD,IH1)
      CALL DMSET(ICMP(5),IBD,IV1)
      CALL DMSET(ICMP(6),IBD,IH2)
      CALL DMSET(ICMP(7),IBD,IV2)
C * * RE-POSITION TRACKING CROSS AND RETURN CONTROL TO CONSOLE
      CALL GITCON(NCOM,0,0)
      CALL AETSKR
   97 FORMAT(*1 NUMBER OF LINES EXCEEDS 40*)
   98 FORMAT(*1 ALL 80 BEAD IDS BEING USED*)
      END
      OVERLAY(4,0)
      PROGRAM CIRC
C * * THIS TASK OVERLAY IS USED FOR DISPLAYING CIRCLE CN CRT
      COMMCN IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
C * * RETRIEVE T.C. COORDINATES OF CENTER OF CIRCLE AND RE-PCSITION
      CALL GITCOF(NCON,IHC,IVC)
      CALL GITCON(NCON,0,0)
C * * DISPLAY CENT BUTTON  AT CENTER OF CIRCLE
      NBYTE=0
      CALL GLRSET(IHC-50,IVC,103B,IBUF,NBYTE,312)
      CALL GLAN(4HCENT,4,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDPBUT,30B,10)
C * * END JOB IF NUMBER OF CIRCLES DISPLAYED IS GREATER THAN 40
      NCIR= NCIR+1
      IF(NCIR.GE.0.AND.NCIR.LE.40) GO TO 10
      PRINT 99
    8 CALL AETSKC(4LQUIT)
C * * WAIT FCR CENT BUTTON TO BE PICKED, THEN ERASE BUTTCN
   10 CALL GIBUT(0,NCON,IDT,IDC)
      IF (IDT.NE.30B.OR.IDC.NE.10) GO TO 10
      CALL GIERAS(IDPBUT)
C * * RETRIEVE T.C. COORDINATES DEFINING RADIUS OF CIRCLE
      CALL GITCOF(NCON,IH2,IV2)
C * * GET CIRCLE BEAD AND STORE BEAD ADDRESS IN ID ARRAY
      CALL DMGTBD(6,IBD)
      DO 12 IDN=1,80
      IF(ID(IDN).EQ.0)GO TO 15
12    CONTINUE
      PRINT 98
      GC TC 8
   15 ID(IDN)=IBD
C * * DISPLAY CIRCLE AND STORE COORDINATES AND DISPLAY ADDRESS IN BEAD
      NBYTE=0
      CALL GLRSET(IH2,IV2,102B,IBUF,NBYTE,312)
      CALL GLARCG(1,IHC,IVC,IH2,IV2,IH2,IV2,-0,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,22B,2,IDN)
      CALL DMSET(ICMP(3),IBD,IDDAD)
      CALL DMSET(ICMP(1),IBD,1)
      CALL DMSET(ICMP(24),IBD,IHC)
      CALL DMSET(ICMP(25),IBD,IVC)
      CALL DMSET(ICMP(4),IBD,IH2)
      CALL DMSET(ICMP(9),IBD,IV2)
      CALL DMSET(ICMP(14),IBD,IH2)
      CALL DMSET(ICMP(19),IBD,IV2)
C * * RE-PCSITION TRACKING CROSS AND RETURN CONTROL TC CCNSCLE
      CALL GITCON(NCON,0,0)
      CALL AETSKR
   98 FORMAT(*1 ALL 80 BEAD IDS BEING USED*)
   99 FORMAT(*1 NUMBER OF CIRCLES EXCEEDS 40*)
      END
      OVERLAY(5,0)
      PROGRAM ERAS
C * * THIS TASK OVERLAY ERASES A LINE OR CIRCLE FROM CRT
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
C * * DETERMINE WHICH LINE OR CIRCLE WAS PICKED FOR ERASURE
      CALL GIFID(NCON,IDT,IDC,IDA)
      IF(IDT.NE.22B.OR.IDC.LE.0.OR.IDC.GE.3) GO TO 25
      IBD=ID(IDA)
      CALL DMGET(ICMP(3),IBD,IDDAD)
      GO TO (10,15)IDC
C * * LINE TO BE ERASED - DECREMENT LINE COUNTER
   10 NLIN=NLIN-1
      ITEMS = NLIN + NCIR
      GO TO 20
C * * CIRCLE TO BE ERASED - DECREMENT CIRCLE COUNTER
   15 NCIR=NCIR-1
      ITEMS = NCIR + NLIN
C * * ERASE THE PICKED DISPLAY ITEM AND RELEASE BEAD
   20 CALL GIERAS(IDDAD)
      CALL DMRLBD(IBD)
      ID(IDA)=0
      IF(ITEMS.GE.0.AND.ITEMS.LE.80) GO TO 25
      CALL AETSKC(4LQUIT)
   25 CALL AETSKR
      END


      OVERLAY (6,0)
      PROGRAM MOVE
C * * THIS TASK OVERLAY MOVES A PICKED DISPLAY ITEM TO A DIFFERENT LOCATION
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION ICRD(5)
C * * RETRIEVE T.C. COORDINATES OF NEW LOCATION
      CALL GITCOF(NCON,IHN,IVN)
C * * DETERMINE WHICH LINE OR CIRCLE IS TO BE MOVED
      CALL GIFID(NCON,IT,IC,IA)
      IF(IT.NE.22B.OR.IC.LE.0.OR.IC.GE.3) GO TO 20
      IBD=ID(IA)
      CALL DMGET(ICMP(3),IBD,IDDAD)
C * * MOVE DISPLAY ITEM
      CALL GIMOVE(IHN,IVN,102B,IDDAD,IT,IC,IA)
      GO TO(10,15)IC
C * * A LINE WAS MOVED - ALTER DISPLAY COORDINATES
   10 DO 11 I=4,7
   11 CALL DMGET(ICMP(I),IBD,ICRD(I-3))
      ICRD(3)=(ICRD(3)-ICRD(1))+IHN
      ICRD(4)=(ICRD(4)-ICRD(2))+IVN
      ICRD(1)=IHN
      ICRD(2)=IVN
      DO 12 I=4,7
   12 CALL DMSET(ICMP(I),IBD,ICRD(I-3))
      GO TO 20
C * * A CIRCLE WAS MOVED - ALTER DISPLAY COORDINATES
   15 CALL DMGET (ICMP(1),IBD,KSHOW)
      CALL DMGET(ICMP(24),IBD,IHC)
      CALL DMGET(ICMP(25),IBD,IVC)
      CALL DMGET(ICMP(4),IBD,IH1)
      CALL DMGET(ICMP(9),IBD,IV1)
      NIHC=(IHN-IH1)+IHC
      NIVC=(IVN-IV1)+IVC
      NREF=NIHC
      IREF=IHC
      CALL DMSET(ICMP(24),IBD,NIHC)
      CALL DMSET(ICMP(25),IBD,NIVC)
      DO 19 J=3,18,5
      DO 17 K=1,KSHOW
      CALL DMGET(ICMP(J+K),IBD,IVAL)
      NVAL=(IVAL-IREF)+NREF
   17 CALL DMSET(ICMP(J+K),IBD,NVAL)
      IF(NREF.EQ.NIHC.AND.IREF.EQ.IHC)GO TO 18
      NREF=NIHC
      IREF=IHC
      GO TO 19
   18 NREF=NIVC
      IREF=IVC
   19 CONTINUE
C * * RE-POSITION TRACKING CROSS AND RETURN CONTROL TO CONSOLE
   20 CALL GITCON(NCON,0,0)
      CALL AETSKR
      END


      OVERLAY(7,0)
      PROGRAM COPY
C * * THIS TASK OVERLAY COPIES A LINE OR CIRCLE ALREADY DISPLAYED
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION ICRD(4)
C * * RETRIEVE T.C. COORDINATES OF LOCATION TO DUPLICATE LINE OR CIRCLE
      CALL GITCOF(NCON,IHN,IVN)
C * * DETERMINE WHICH ITEM WAS PICKED FOR COPY AND RETRIEVE ITS DISPLAY
C * * ADDRESS FROM BEAD
      CALL GIFID(NCON,IT,IC,IDN)
      IF(IT.NE.22B.OR.IC.LE.0.OR.IC.GE.3)GO TO 25
      IA=ID(IDN)
      CALL DMGET(ICMP(3),IA,IPDAD)
C * * ASCERTAIN WHETHER THERE IS ROOM FOR ANOTHER BEAD
      DO 5 IDN=1,80
      IF(ID(IDN).EQ.0)GO TO 7
5     CONTINUE
      GO TO 22
    7 GO TO(10,15)IC
C * * COPYING LINE - GET NEW LINE BEAD
   10 CALL DMGTBD(2,IBD)
      ID(IDN)=IBD
      DO 11 I=4,7
   11 CALL DMGET(ICMP(I),IA,ICRD(I-3))
C * * DETERMINE DISPLAY COORDINATES OF DUPLICATE LINE AND STORE IN BEAD
      ICRD(3)=(ICRD(3)-ICRD(1))+IHN
      ICRD(4)=(ICRD(4)-ICRD(2))+IVN
      ICRD(1)=IHN
      ICRD(2)=IVN
      DO 12 I=4,7
   12 CALL DMSET(ICMP(I),IBD,ICRD(I-3))
      NLIN=NLIN+1
      ITEMS = NLIN + NCIR
      GO TO 20
C * * COPYING CIRCLE - GET NEW CIRCLE BEAD
   15 CALL DMGTBD(6,IBD)
      ID(IDN)=IBD
```

```
C * * DETERMINE DISPLAY COORDINATES OF DUPLICATE CIRCLE AND STORE IN READ
      CALL DMGET(ICMP(1),IA,KSHOW)
      CALL DMGET(ICMP(24),IA,IHC)
      CALL DMGET(ICMP(25),IA,IVC)
      CALL DMGET(ICMP(4),IA,IH1)
      CALL DMGET(ICMP(9),IA,IV1)
      NIHC=(IHN-IH1)+IHC
      NIVC = (IVN - IV1) + IVC
      NREF=NIHC
      IREF=IHC
      CALL DMSET(ICMP(1),IBD,KSHOW)
      CALL DMSET(ICMP(24),IBD,NIHC)
      CALL DMSET(ICMP(25),IBD,NIVC)
      DO 19 J=3,18,5
      DO 17 K=1,KSHOW
      CALL DMGET(ICMP(J+K),IA,IVAL)
      NVAL = (IVAL - IREF) + NREF
   17 CALL DMSET(ICMP(J+K),IBD,NVAL)
      IF(NREF.EQ.NIHC.AND.IREF.EQ.IHC)GO TO 18
      NREF=NIHC
      IREF=IHC
      GO TO 19
   18 NREF=NIVC
      IREF=IVC
   19 CONTINUE
      NCIR=NCIR+1
      ITEMS = NCIR + NLIN
   20 IF(ITEMS.GE.0.AND.ITEMS.LE.80)GO TO 23
   22 CALL AETSKC (4LQUIT)
C * * COPY DISPLAY ITEM, STORE DISPLAY ADDRESS, AND RE-POSITION TR. CROSS
   23 CALL GICOPY(IDDAD,NCON,IHN,IVN,102B,IDDAD,IT,IC,IDN)
      CALL DMSET(ICMP(3),IBD,IDDAD)
   25 CALL GITCON(NCON,0,1)
      CALL AETSKR
      END


      OVERLAY(10,0)
      PROGRAM ZOOM
C * * THIS TASK OVERLAY ZOOMS UP OR DOWN THE ENTIRE DISPLAY
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION IBCD(3),IDR(4)
C * * STORE ZOOM BUTTONS, T.C. COORDINATES ARE FOR CENTER OF FRAME
      IBCD(1)=7HZOOM UP
      IBCD(2)=9HZOOM DOWN
      IBCD(3)=4HCENT
      CALL GITCOF(NCON,IHCEN,IVCEN)
      CALL GITCON(NCON,0,0)
      K=1
C * * DISPLAY ZOOM CHOICES AND CENT AT CENTER OF FRAME
      IH=-200
      IV=-1500
    5 NBYTE=0
      CALL GLRSET(IH,IV,103B,IBUF,NBYTE,312)
      CALL GLAN(IBCD(K),9,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDR(K),44B,K)
      IV=IV-200
      K=K+1
      GO TO (5,5,7,9)K
    7 IH=IHCEN-50
      IV=IVCEN
      GO TO 5
C * * DEFINE ZOOM UP AND ZOOM DOWN AS PRIME BUTTONS
    9 CALL GIPBUT(NCON,44B,30B,44B)
C * * DISPLAY NUMERIC FONT AND LIGHT REGISTER
      CALL GFONTN(NCON,-1300,-500,IDR(4))
      CALL GIANS(NCON,10,-1500,-900)
C * * WAIT FOR ZOOM UP OR ZOOM DOWN TO BE PICKED
   10 CALL GIBUT(0,NCON,IT,IC)
      IF(IT.NE.30B.AND.IC.NE.44B)GO TO 10
      N=1
      CALL GIFSID(NCON,N,IT,IC)
      IF(IT.NE.44B.OR.IC.LE.0.OR.IC.GE.3)GO TO 10
C * * RETRIEVE T.C. COORDINATES OF UPPER RT. HAND CORNER OF FRAME
      CALL GITCOF(NCON,IHCOR,IVCOR)
      NC=10
C * * RETRIEVE ZOOM FACTOR FROM LIGHT REGISTER AND ERASE ZOOM MESSAGES
      CALL GIANE(NCON,NC,IBCD)
      CALL GIERAS(IDR(1),IDR(2),IDR(3),IDR(4))
      ZM=1.
      IF(NC.LE.0.OR.NC.GE.11) GO TO 11
 2001 FORMAT(2H(E,I2,3H.1))
      ENCODE(7,2001,SMAT)NC
      DECODE(NC,SMAT,IBCD)ZM
C * * ERASE ALL CURRENT DISPLAY ITEMS FROM CRT
   11 DO 20 J=1,80
      IF (ID(J)) 20,20,12
   12 CALL DMGET(ICMP(1),ID(J),KSHOW)
      CALL DMGET(ICMP(3),ID(J),IDDAD)
      CALL GIERAS(IDDAD)
      IF(KSHOW) 20,15,16
C * * FRAME SCISSOR ANY LINES OR CIRCLES NOT WITHIN FRAME BOUNDARIES
   15 CALL LINSCIS(IC,J,IHCEN,IVCEN,IHCOR,IVCOR,ZM)
      GO TO 20
   16 CALL ARCSCIS(IC,J,IHCEN,IVCEN,IHCOR,IVCOR,ZM)
   20 CONTINUE
      CALL GITCON(NCON,0,0)
      CALL AETSKR
      END


      SUBROUTINE LINSCIS(K,J,IHCEN,IVCEN,IHCOR,IVCOR,ZM)
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION ICRD(5)
      DO 5 I=4,7
    5 CALL DMGET(ICMP(I),ID(J),ICRD(I-3))
      GO TO (10,11)K
C * * ALTER LINE DISPLAY COORDINATES BY ZOOM FACTOR(DIVISOR)
   10 H1=ICRD(1)*ZM
      V1=ICRD(2)*ZM
      H2=ICRD(3)*ZM
      V2=ICRD(4)*ZM
      GO TO 15
   11 H1=ICRD(1)/ZM
      V1=ICRD(2)/ZM
      H2=ICRD(3)/ZM
      V2=ICRD(4)/ZM
```

```
C * * FRAME SCISSOR LINE TO BE DISPLAYED
   15 CALL GLLINE(IHCEN,IVCEN,IHCOR,IVCOR,H1,V1,H2,V2,KSHOW,
     *ICRD(1),ICRD(2),ICRD(3),ICRD(4))
      IF(KSHOW)17,17,18
   17 NLIN=NLIN-1
C * * RELEASE BEAD IF ENTIRE LINE OUTSIDE OF FRAMD
      CALL DMRLBD(ID(J))
      IF(NLIN.GE.0.AND.NLIN.LE.40)GO TO 20
      PRINT 97
      CALL AETSKC(4LQUIT)
C * * DISPLAY ZOOMED LINE AND STORE NEW DISPLAY COORDINATES AND ADDRESS IN BEAD
   18 NBYTE=0
      CALL GLRSET(ICRD(1),ICRD(2),102B,IBUF,NBYTE,312)
      CALL GLSEGS(ICRD(1),ICRD(2),ICRD(3),ICRD(4),1,0,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,22B,1,J)
      CALL DMSET(ICMP(3),ID(J),IDDAD)
      DO 19 I=4,7
   19 CALL DMSET(ICMP(I),ID(J),ICRD(I-3))
   20 RETURN
   97 FORMAT(*1 NUMBER OF LINES EXCEEDS 40*)
      END


      SUBROUTINE ARCSCIS(K,J,IHCEN,IVCEN,IHCOR,IVCOR,ZM)
C * * DO SAME TASKS FOR CIRCLES (ARCS) AS WAS DONE FOR LINES IN SUB. LINSCIS
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION IH1(5),IV1(5),IH2(5),IV2(5)
      CALL DMGET(ICMP(24),ID(J),IHC)
      CALL DMGET(ICMP(25),ID(J),IVC)
      CALL DMGET(ICMP(4),ID(J),IH1)
      CALL DMGET(ICMP(9),ID(J),IV1)
      GO TO (11,12)K
   11 HC=IHC*ZM
      VC=IVC*ZM
      H1=IH1*ZM
      V1=IV1*ZM
      GO TO 15
   12 HC=IHC/ZM
      VC=IVC/ZM
      H1=IH1/ZM
      V1=IV1/ZM
   15 CALL GLARC(IHCEN,IVCEN,IHCOR,IVCOR,HC,VC,H1,V1,H1,V1,KSHOW,
     *IHC,IVC,IH1,IV1,IH2,IV2)
      IF(KSHOW)17,17,18
   17 CALL DMRLBD(ID(J))
      NCIR=NCIR-1
      IF(NCIR.GE.0.AND.NCIR.LE.40)GO TO 25
      PRINT 99
      CALL AETSKC(4LQUIT)
   18 DO 20 I=1,KSHOW
      CALL DMSET(ICMP(3+I),ID(J),IH1(I))
      CALL DMSET(ICMP(8+I),ID(J),IV1(I))
      CALL DMSET(ICMP(13+I),ID(J),IH2(I))
   20 CALL DMSET(ICMP(18+I),ID(J),IV2(I))
      CALL DMSET (ICMP(24),ID(J),IHC)
      CALL DMSET (ICMP(25),ID(J),IVC)
      CALL DMSET(ICMP(1),ID(J),KSHOW)
      NBYTE=0
      CALL GLRSET(IH1(1),IV1(1),102B,IBUF,NBYTE,312)
      CALL GLARCG(KSHOW,IHC,IVC,IH1,IV1,IH2,IV2,0,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,22B,2,J)
      CALL DMSET(ICMP(3),ID(J),IDDAD)
   25 RETURN
   99 FORMAT(*1 NUMBER OF CIRCLES EXCEEDS 40*)
      END


      OVERLAY(11,0)
      PROGRAM ROTA
C * * THIS TASK OVERLAY ROTATES CRT DISPLAY UNTIL STOPPED OR 360 DEGREES
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      COMMON /BLOCKA/IORIG
      RINC = 3.1416 / 30.0
      R = RINC
      IORIG = 2
C * * DISPLAY STOP BUTTON
      NBYTE=0
      CALL GLRSET(-200,-1500,103B,IBUF,NBYTE,312)
      CALL GLAN(4HSTOP,4,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDSTOP,30B,10)
C * * DISPLAY GO BUTTON
      NBYTE = 0
      CALL GLRSET(-200,-1600,103B,IBUF,NBYTE,312)
      CALL GLAN(2HGO,2,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDGO,30B,99)
C * * DISPLAY PROMPT MESSAGE AND FONT
      NBYTE = 0
      CALL GLRSET(-1500,-1000,3B,IBUF,NBYTE,312)
      CALL GLAN(48HINPUT ANGLE TO BE ROTATED THROUGH - THEN PICK GO,48,
     1   IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDMESS,0,98)
      CALL GFONTN(NCON,-1300,-500,IFONT)
      CALL GIANS(NCON,10,-1500,-900)
C * * WAIT FOR GO TO BE PICKED
  100 CALL GIBUT(0,NCON,IT,IC)
      IF(IC.NE.99)  GO TO 100
      NC = 10
      CALL GIANE(NCON,NC,IBCD)
      CALL GIERAS(IFONT,IDMESS)
      IF(NC.LE.0)  GO TO 11
 2001 FORMAT(2H(E,I2,3H.1))
      ENCODE(7,2001,SMAT) NC
      DECODE(NC,SMAT,IBCD) ENDANG
      GO TO 12
   11 ENDANG = 361.0
C * * ROTATE DISPLAY ELEMENTS IN 6 DEGREE INCREMENTS
   12 DO 15 I = 1 , 360 , 6
      DO 10 J=1,80
      IF(ID(J))10,10,5
C * * RETRIEVE DISPLAY ITEM TYPE AND ASSOCIATIVE ADDRESS, AND ERASE OLD DISPLAY
    5 CALL DMGET(ICMP(1),ID(J),K)
      CALL DMGET(ICMP(3),ID(J),IDDAD)
      CALL GIERAS(IDDAD)
C * * DISPLAY LINE(CIRCLE) ROTATED 6 DEGREES
      IF(K)10,7,8
    7 CALL LINROTA(J,RINC,I)
      GO TO 10
    8 CALL ARCROTA(J,RINC,I)
   10 CONTINUE
C * * CHECK TO SEE IF ANGLE LIMIT HAS BEEN REACHED
      ANG = I
      IF(ANG.GE.ENDANG)  GO TO 20
C * * INCREMENT ROTATION ANGLE
      R = R + RINC
C * * DETERMINE IF STOP BUTTON PICKED
      CALL GIBUT(1,NCON,IDT,IDC)
      IF ((IDT .EQ. 30B) .AND. (IDC .EQ. 10)) GO TO 20
   15 CONTINUE
C * * ERASE STOP BUTTON AND RETURN CONTROL TO CONSOLE
   20 CALL GIERAS(IDSTOP)
      IORIG = 1
C * * DISPLAY ORIGINAL DISPLAY BUTTON
      NBYTE = 0
      CALL GLRSET(-200,-1500,103B,IBUF,NBYTE,312)
      CALL GLAN(17HORIGINAL DISPLAY,17,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDORIG,30B,97)
C * * SEE IF GO OR ORIGINAL BUTTON IS PICKED
      CALL GIBUT(0,NCON,IDT,IDC)
      CALL GIERAS(IDORIG,IDGO)
      IF(IDC.EQ.99)  GO TO 99
C * * RETURN TO ORIGINAL DISPLAY
      I = 2
      DO 30  J=1,80
      IF(ID(J))  30,30,50
C * * RETRIEVE DISPLAY ITEM TYPE AND ASSOCIATIVE ADDRESS, AND ERASE OLD DISPLAY
   50 CALL DMGET(ICMP(1),ID(J),K)
      CALL DMGET(ICMP(3),ID(J),IDDAD)
      CALL GIERAS(IDDAD)
C * * DISPLAY ORIGINAL LINE OR CIRCLE
      IF(K)  30,70,80
   70 CALL LINROTA(J,RINC,I)
      GO TO 30
   80 CALL ARCROTA(J,RINC,I)
   30 CONTINUE
   99 CALL AETSKR
      END


      SUBROUTINE LINROTA(J,THETA1,K)
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION ICRD(5)
      DO 5 I=4,7
    5 CALL DMGET(ICMP(I),ID(J),ICRD(I-3))
C * * CHANGE LINE DISPLAY COORDINATES TO ROTATED VALUES
      L = 1
      CALL ANGLE(ICRD(1),ICRD(2),THETA1,K,J,L)
      L = 2
      CALL ANGLE(ICRD(3),ICRD(4),THETA1,K,J,L)
C * * DISPLAY NEW LINE
      NBYTE=0
      CALL GLRSET(ICRD(1),ICRD(2),102B,IBUF,NBYTE,312)
      CALL GLSEGS(ICRD(1),ICRD(2),ICRD(3),ICRD(4),1,-0,IBUF,NBYTE,312)
      CALLGIDISP(NCON,IBUF,NBYTE,IDDAD,22B,1,J)
C * * STORE NEW DISPLAY COORDINATES
      CALL DMSET(ICMP(3),ID(J),IDDAD)
      CALL GLARCG(KSHOW,IHC,IVC,IH1,IV1,IH2,IV2,0,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,22B,2,J)
      CALL DMSET(ICMP(3),ID(J),IDDAD)
      RETURN
      END


      SUBROUTINE ANGLE(IH,IV,THETA1,K,J,L)
C
C    ROTATES A LINE FROM (0,0) TO (IH,IV) THROUGH THETA1 DEGREES IN
C    A COUNTERCLOCKWISE DIRECTION BY ROTATING THE (IH,IV) COORDINATES.
C
      COMMON /BLOGKA/IORIG
      TH = IH
      TV = IV
      D = SQRT(TH**2 + TV**2)
      IF((ABS(D)-0.01).LT.0.0)   RETURN
      THETA2 = ATAN2(TV,TH)
      IF((IORIG.EQ.1).AND.(J.EQ.1))  THETF = THETA2
      IF(IORIG.EQ.1)  GO TO 20
C * * STORE REFERENCE ANGLE OF FIGURE FOR LATER USE TO RETURN TO OLD DISPLAY
      IF((K.EQ.1).AND.(L.EQ.1).AND.(J.EQ.1)) THETI = THETA2
      THETA2 = THETA2 + THETA1
   10 IV = D*SIN(THETA2)
      IH = D*COS(THETA2)
      RETURN
C * * COMPUTE ANGLE CHANGE NECESSARY TO PUT ITEM IN ORIGIONAL POSITION
   20 THETA2 = 6.28318 - THETF + THETI + THETA2
      GO TO 10
      END


      SUBROUTINE ARCROTA(J,THETA1,K)
C * * SAME TASK AS SUB. LINROTA EXCEPT THIS ONE FOR CIRCLES(ARCS)
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION IH1(5),IV1(5),IH2(5),IV2(5)
      CALL DMGET(ICMP(1),ID(J),KSHOW)
      CALL DMGET(ICMP(24),ID(J),IHC)
      CALL DMGET(ICMP(25),ID(J),IVC)
      L = 1
      CALL ANGLE(IHC,IVC,THETA1,K,J,L)
      L = 2
      CALL DMSET(ICMP(24),ID(J),IHC)
      CALL DMSET(ICMP(25),ID(J),IVC)
      DO 10 I=1,KSHOW
      CALL DMGET(ICMP(3+I),ID(J),IH1(I))
      CALL DMGET(ICMP(8+I),ID(J),IV1(I))
      CALL ANGLE(IH1(I),IV1(I),THETA1,K,J,L)
      CALL DMSET(ICMP(3+I),ID(J),IH1(I))
      CALL DMSET(ICMP(8+I),ID(J),IV1(I))
      CALL DMGET(ICMP(13+I),ID(J),IH2(I))
      CALL DMGET(ICMP(18+I),ID(J),IV2(I))
      CALL ANGLE(IH2(I),IV2(I),THETA1,K,J,L)
      CALL DMSET(ICMP(13+I),ID(J),IH2(I))
   10 CALL DMSET(ICMP(18+I),ID(J),IV2(I))
      NBYTE=0
      CALL GURSET(IH1(1),IV1(1),102B,IBUF,NBYTE,312)
      CALL DMSET(ICMP(4),ID(J),ICRD(1))
      CALL DMSET(ICMP(5),ID(J),ICRD(2))
      CALL DMSET(ICMP(6),ID(J),ICRD(3))
      CALL DMSET(ICMP(7),ID(J),ICRD(4))
      RETURN
      END



      OVERLAY(12,0)
      PROGRAM QUIT
C * * THIS TASK OVERLAY CALLED TO TERMINATE JOB
      COMMON IBUF(64),NCON,ICMP(25),ID(80),NLIN,NCIR
      DIMENSION NAME(2)
      DATA NAME /10HJOB QUIT**,10H SO LONG**/
C * * DISPLAY END MESSAGE AND RELEASE CONSOLE
      NBYTE=0
      CALL GLRSET(0,1500,102B,IBUF,NBYTE,312)
      CALL GLAN(NAME,20,IBUF,NBYTE,312)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD)
      CALL GICNRL(NCON)
      STOP
      END
```

```
7
 8
  9
KCB
SCR
7
 8
  9
KCB
7
 8
  9

6
 7
  8
   9
```

SAMPLE CODING PROBLEMS

Sample code listings for the Winding Road, Drifting Circle, Menu, Scissored Arc, and Star of David problems are given below. The examples given represent but one method of coding the problem. Other methods, perhaps more efficient or better-suited to specific installation requirements, may easily be adopted. These examples differ from those given in the text because they are written in FORTRAN Extended under Version 2 rather than RUN FORTRAN under Version 1, they were run using console (NCON) 9, not console 1, they contain additional coding to check errors, and they demonstrate efficient programming techniques. The IGS Version 2 permanent files option is not used by the examples in the text or in this appendix.

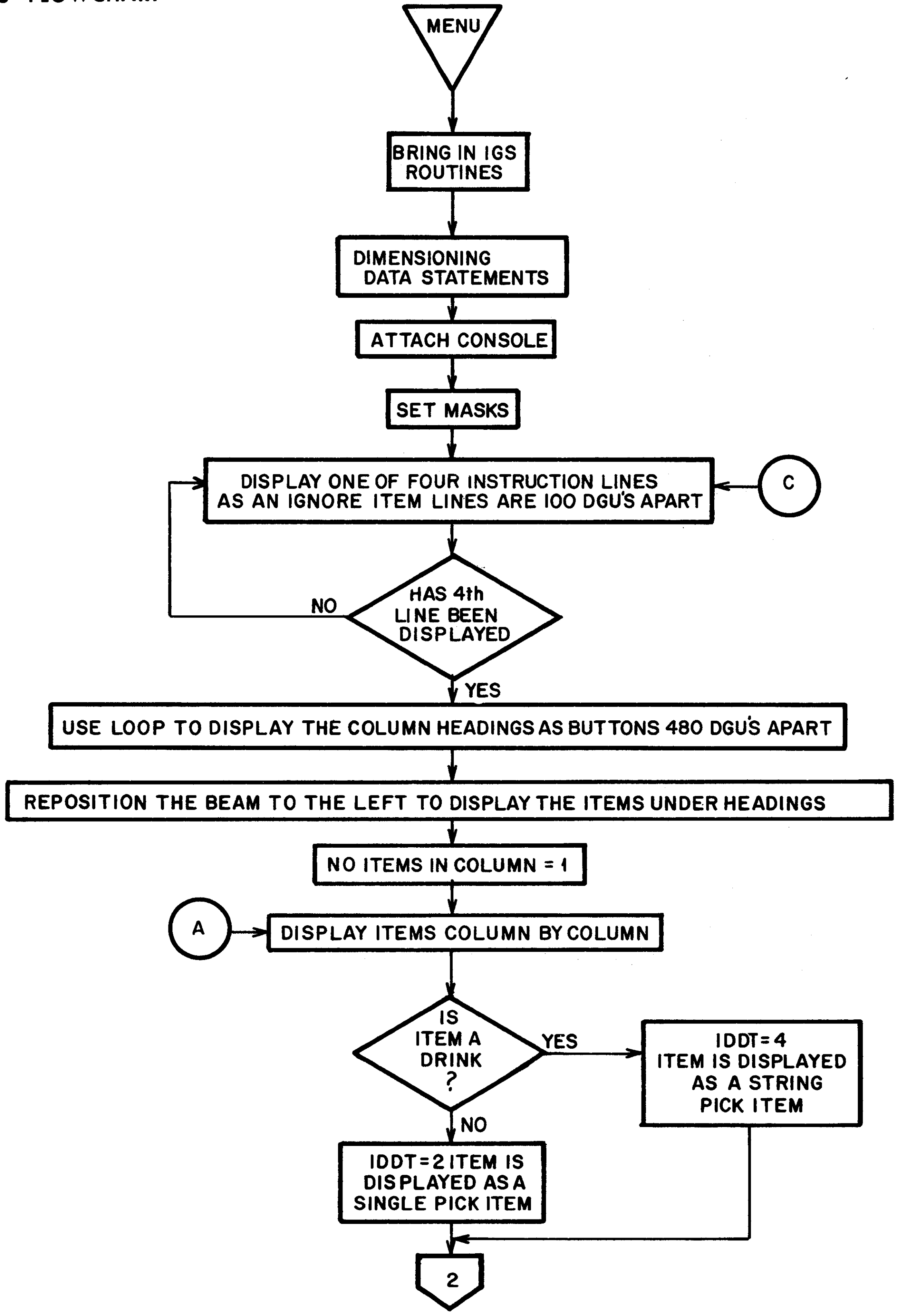
WINDING ROAD

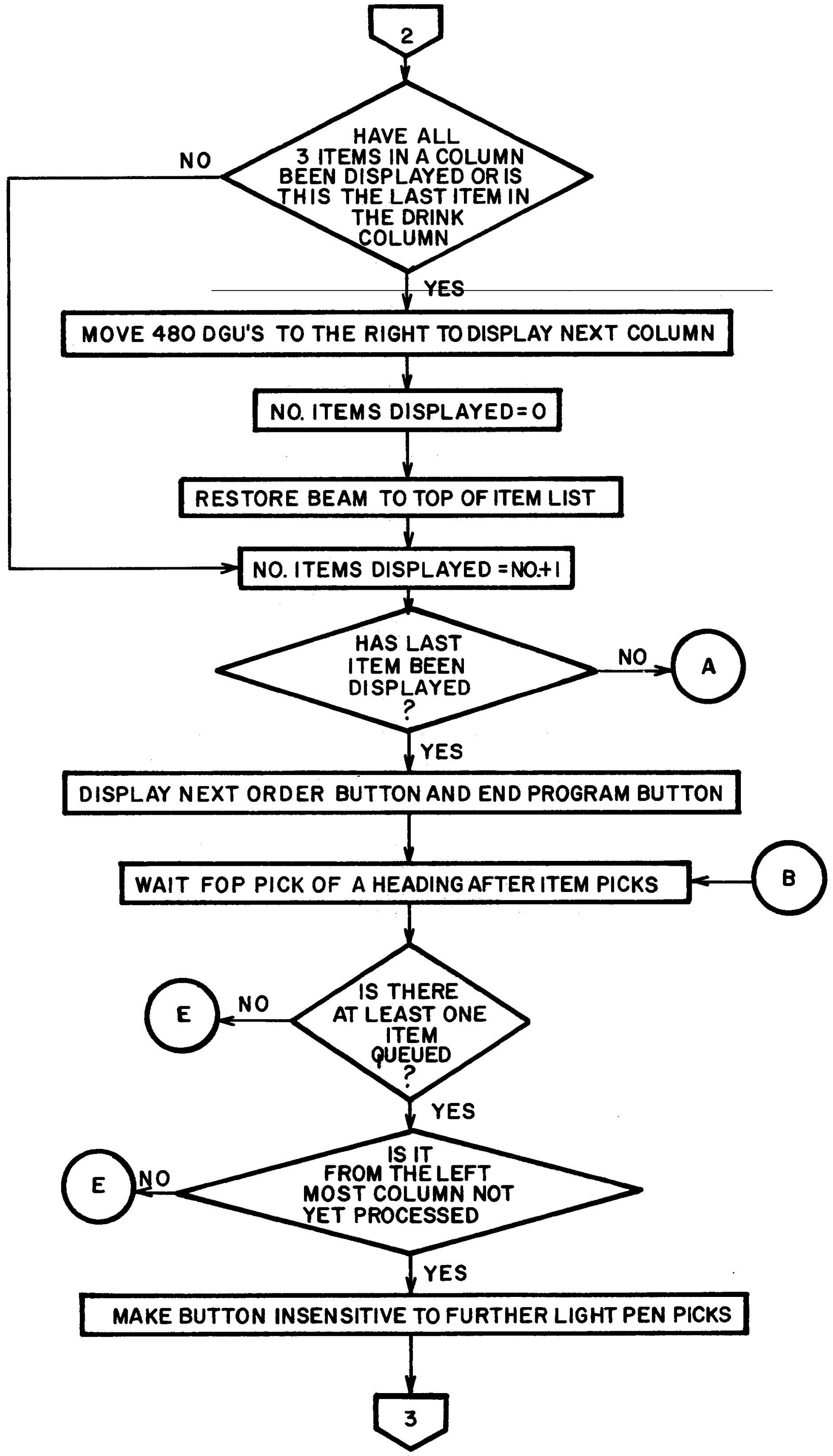
```
      OVERLAY(SOURCE,0,0)
      PROGRAM CREATE
      CALL AEXEC
      END
      OVERLAY(1,0)
      PROGRAM JSLOW
      DIMENSION IBUF(64),IBCD(3)
      NCON=9
      NBYTE=0
      MBYTE=310
C   ATTACH CONSOLE AND CLEAR BUFFERS
      CALL GICNJB(NCON)
      CALL GIMASK(NCON,0,8,8)
C   DISPLAY DIAMOND SHAPED PERIMETER
5     CALL GLRSET(0,1500,102B,IBUF,NBYTE,MBYTE)
      CALL GLSEGS(0,1500,1500,0,1,7777B,IBUF,NBYTE,MBYTE)
      CALL GLSEG(0,-1500,1)
      CALL GLSEG(-1500,0,1)
      CALL GLSEG(0,1500,1)
      CALL GIDISP(NCON,IBUF,NBYTE,ISQR)
C   DISPLAY *WINDING ROAD* LABEL
      CALL GLRSET(-84,770,2,IBUF,NBYTE,MBYTE)
      CALL GLAN(7HWINDING,7,IBUF,NBYTE,MBYTE)
      CALL GLRSET(-48,-770,2,IBUF,NBYTE,MBYTE)
      CALL GLAN(4HROAD,4,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ILBL)
      CALL GFONTA(NCON,1500,1500,IA,IB)
C   DISPLAY CURVE
      CALL GLRSET(256,256,102B,IBUF,NBYTE,MBYTE)
      CALL GLARCG(1,0,256,256,256,0,0,7777B,IBUF,NBYTE,MBYTE)
      CALL GLRSET(-256,-256,102B,IBUF,NBYTE,MBYTE)
      CALL GLARCG(1,0,-256,-256,-256,0,0,7777B,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IS)
      IF(I.GT.0)GO TO 10
C   DISPLAY *ERASE CURVE* BUTTON
      CALL GLRSET(-1280,-900,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(11HERASE CURVE,11,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IECRV,8,1)
C   DISPLAY *DISPLAY SIGN AGAIN* BUTTON
      CALL GLRSET(-1280,-1000,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(18HDISPLAY SIGN AGAIN,18,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDISA,8,2)
C    DISPLAY *END PROGRAM* BUTTON
      CALL GLRSET(-1280,-1100,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(11HEND PROGRAM,11,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IEND,8,3)
C    WAIT FOR A BUTTON PICK
10    CALL GICLR(NCON)
      CALL GIBUT(0,NCON,IDDT,IDDC)
      IF(IDDC.GT.1)GO TO 20
C    ERASE CURVE
      CALL GIERAS(IS)
      GO TO 10
20    IF(IDDC.GT.2)GO TO 30
C     ERASE SIGN
      CALL GIERAS(ISQR,ILBL,IS)
C    BUTTONS ALREADY DISPLAYED
      I=1
C    DISPLAY SIGN AGAIN
      GO TO 5
30    CALL GLRSET(-190,1600,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(16HPROGRAM IS ENDED,16,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ISTOP)
      STOP
      END
```

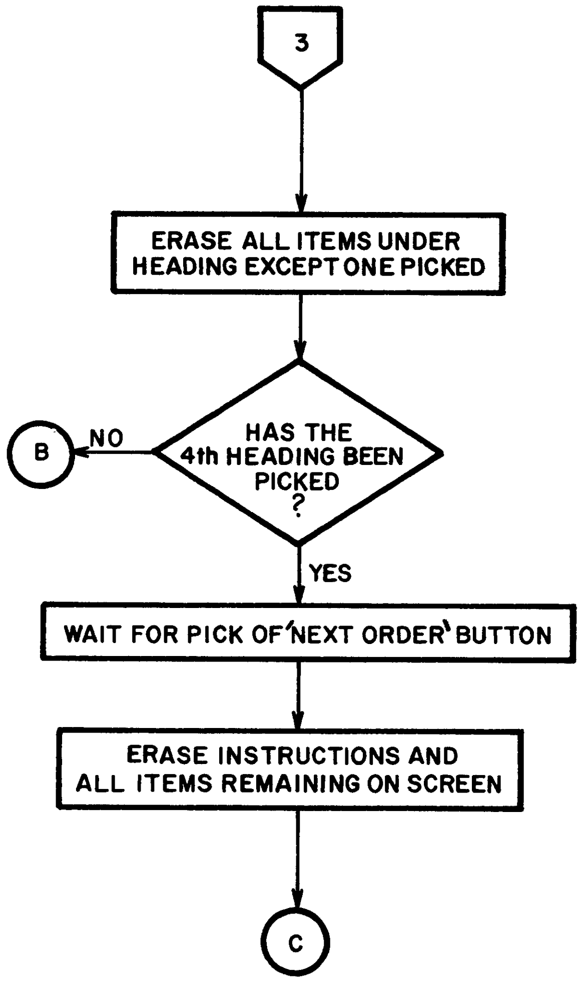
DRIFTING CIRCLE

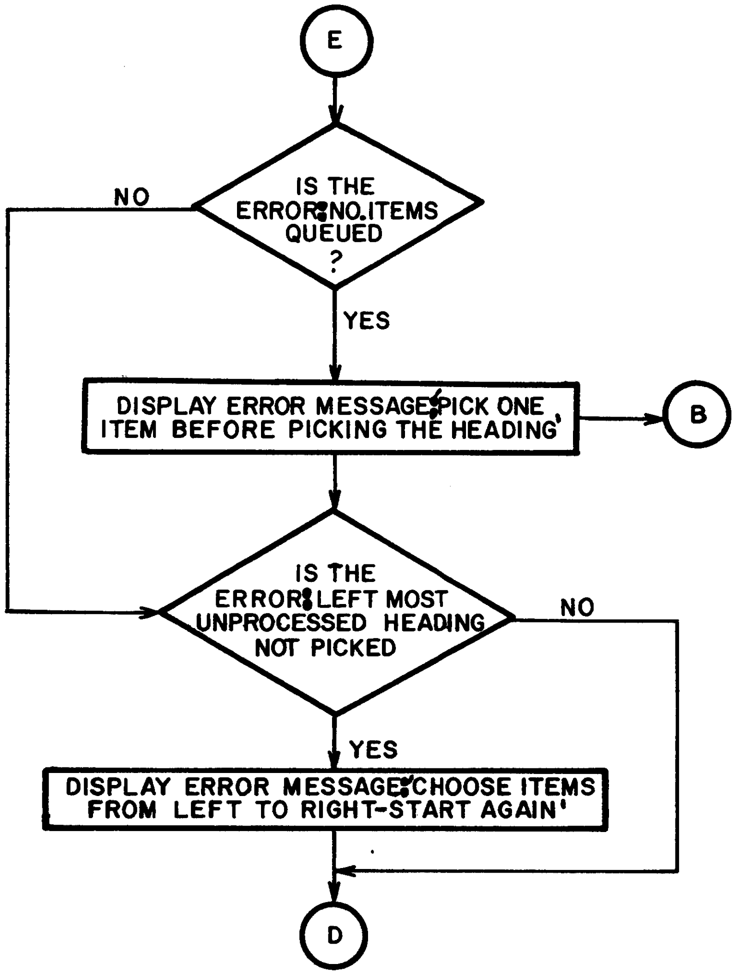
```
      OVERLAY(SOURCE,0,0)
      PROGRAM CREATE
      CALL AEXEC
      END


      OVERLAY(1,0)
      PROGRAM DRIFT
      DIMENSION IBUF(64),IBCD(3)
      NCON=9
C    ATTACH CONSOLE
      CALL GICNJB(NCON)
      MBYTE=310
      NBYTE=0
C    SET MASKS
      DO 100 I=1,5
      N=2**(I-1)
100   CALL GIMASK(NCON,-0,N,N)
C    DISPLAY *REPEAT* BUTTON WHICH BLINKS WHEN PICKED
101   CALL GLRSET(-70,-1200,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(6HREPEAT,6,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IRPT,8+16,2)
C    DISPLAY *COPY CIRCLE BUTTON*WHICH BLINKS WHEN PICKED
      CALL GLRSET(-70,-1400,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(11HCOPY CIRCLE,11,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ICPY,8+16,1)
C    DISPLAY *END PROGRAM* BUTTON WHICH BLINKS WHEN PICKED
      CALL GLRSET(-70,-1600,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(11HEND PROGRAM,11,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IEND,8+16,3)
      L=0
C   DISPLAY TITLE
      CALL GURSET(-180,1000,2,IBUF,NBYTE,MBYTE)
      CALL GUAN(15HDRIFTING CIRCLE,15,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ILBL)
C   DISPLAY CIRCLE MACRO
      CALL GUARCG(1,-1024,0,-512,0,-512,0,7777B,IBUF,NBYTE,MBYTE)
      CALL GIMAC(NCON,IBUF,NBYTE,MCIRC)
      CALL GURSET(-512,0,102B,IBUF,NBYTE,MBYTE)
      CALL GUMACG(MCIRC,1,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ITEM1)
C   TURN TRACKING CROSS ON
      CALL GITCON(NCON,-512,0)
C   ATTACH MACRO TO TRACKING CROSS
      CALL GITMMV(NCON,MCIRC)
C   ACTIVATE LIGHT PEN KEY
      CALL GILPKY(NCON,8,7)
      K=0
202   K=K+1
201   CALL GICLR(NCON)
      CALL GIBUT(0,NCON,IDDT,IDDC,IDWA,IDWB,IH,IV)
      IF(IDDC.EQ.1)GO TO 210
      IF(IDDC.GE.2.AND.IDDC.LT.7)GO TO 203
      IF(K.LT.3)GO TO 202
      IF(IDDC.NE.7.AND.L.LT.1) GO TO 201
C   DISPLAY CIRCLE IN NEW LOCATION
210   IF(IH.EQ.-512.AND.IV.EQ.0)GO TO 201
      CALL GURSET(IH,IV,102B,IBUF,NBYTE,MBYTE)
      CALL GUMACG(MCIRC,1,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ITEM2)
      CALL GILPKY(NCON)
      CALL GITMMV(NCON,0)
      L=1
      CALL GIERAS(ITEM1)
C   DISPLAY A COPY OF CIRCLE
      CALL GICOPY(ITEM2,NCON,-300,-100,106B,ITEM3)
      GO TO 201
C   ERASE CONSOLE
203   IF(IDDC.GT.2)GO TO 204
      CALL GIERAS(ILBL,IRPT,IEND,ICPY,ITEM2,ITEM3)
      GO TO 101
204   CALL GURSET(-190,1600,102B,IBUF,NBYTE,MBYTE)
      CALL GUAN(16HPROGRAM IS ENDED,16,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,ISTOP)
      STOP
      END
```

MENU FLOWCHART


MENU
BRING IN IGS ROUTINES
DIMENSIONING DATA STATEMENTS
ATTACH CONSOLE
SET MASKS
DISPLAY ONE OF FOUR INSTRUCTION LINES AS AN IGNORE ITEM LINES ARE 100 DGU'S APART
C
HAS 4th LINE BEEN DISPLAYED
NO
YES
USE LOOP TO DISPLAY THE COLUMN HEADINGS AS BUTTONS 480 DGU'S APART
REPOSITION THE BEAM TO THE LEFT TO DISPLAY THE ITEMS UNDER HEADINGS
NO ITEMS IN COLUMN = 1
A
DISPLAY ITEMS COLUMN BY COLUMN
IS ITEM A DRINK ?
YES
IDDT=4 ITEM IS DISPLAYED AS A STRING PICK ITEM
NO
IDDT=2 ITEM IS DISPLAYED AS A SINGLE PICK ITEM
2


2
HAVE ALL 3 ITEMS IN A COLUMN BEEN DISPLAYED OR IS THIS THE LAST ITEM IN THE DRINK COLUMN
NO
YES
MOVE 480 DGU'S TO THE RIGHT TO DISPLAY NEXT COLUMN
NO. ITEMS DISPLAYED = 0
RESTORE BEAM TO TOP OF ITEM LIST
NO. ITEMS DISPLAYED = NO.+1
HAS LAST ITEM BEEN DISPLAYED ?
NO
A
YES
DISPLAY NEXT ORDER BUTTON AND END PROGRAM BUTTON
WAIT FOP PICK OF A HEADING AFTER ITEM PICKS
B
IS THERE AT LEAST ONE ITEM QUEUED ?
NO
E
YES
IS IT FROM THE LEFT MOST COLUMN NOT YET PROCESSED
NO
E
YES
MAKE BUTTON INSENSITIVE TO FURTHER LIGHT PEN PICKS
3


3
ERASE ALL ITEMS UNDER HEADING EXCEPT ONE PICKED
HAS THE 4th HEADING BEEN PICKED ?
NO
B
YES
WAIT FOR PICK OF 'NEXT ORDER' BUTTON
ERASE INSTRUCTIONS AND ALL ITEMS REMAINING ON SCREEN
C


E
IS THE ERROR: NO ITEMS QUEUED ?
NO
YES
DISPLAY ERROR MESSAGE 'PICK ONE ITEM BEFORE PICKING THE HEADING'
B
IS THE ERROR: LEFT MOST UNPROCESSED HEADING NOT PICKED
NO
YES
DISPLAY ERROR MESSAGE 'CHOOSE ITEMS FROM LEFT TO RIGHT-START AGAIN'
D

```
      OVERLAY(SOURCE,0,0)
      PROGRAM CREATE(INPUT,OUTPUT)
      CALL AEXEC
      END

      OVERLAY(1,0)
      PROGRAM NMENU
      DIMENSION IER(57),IDTX(4),IDCX(4),IBCD(54),IBUF(64)
      DATA(IBCD(I),I=1,13,4)/40HCHOOSE ONE APPETIZER ,THEN HEADING
     1, 40HCHOOSE ONE ENTREE     ,THEN HEADING         ,
     2, 40HCHOOSE ONE DESSERT    ,THEN HEADING         ,
     3  40HCHOOSE DRINKS(.LE.4) ,THEN HEADING         /
      DATA(IBCD(J),J=17,53,2)/20HAPPETIZER                  ,
     120HENTREE               ,20HDESSERT                   ,
     220HDRINK                ,20HCRAB COCKTAIL             ,
     320HTOMATO JUICE         ,20HCHOWDER                   ,
     420HSTEAK                ,20HLOBSTER                   ,
     520HCHICKEN              ,20HSHERBET                   ,
     620HPIE                  ,20HCAKE                      ,
     720HTEA                  ,20HCOFFEE                    ,
     820HMILK                 ,20HWINE                      ,
     920HNEXT ORDER           ,20HEND PROGRAM               /
500   IV=550
      JCK=0
      NCON=9
      NBYTE=0
      MBYTE=310
      CALL GICNJB(NCON)
      DO 10 I=2,5
      J=2**(I-1)
10    CALL GIMASK(NCON,0,J,J)
C  DISPLAY INSTRUCTION LINES
      DO 20 I=1,13,4
      CALL GLRSET(-840,IV,1028,IBUF,NBYTE,MBYTE)
      CALL GLAN(IBCD(I),40,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(I),1,I)
20    IV=IV-100
C  DISPLAY MENU LABELS AS BUTTONS
      IH=-960
      DO 30 J=17,23,2
      CALL GLRSET(IH,50,1038,IBUF,NBYTE,MBYTE)
      CALL GLAN(IBCD(J),13,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(J),24,J)
30    IH=IH+480
      IH=-960
      IV=-50
      IDDT=2
      ICNT=1
C  DISPLAY ALL ITEMS ON THE MENU AS SINGLE PICKS EXCEPT DRINKS
C  DRINKS ARE STRING PICKS
      DO 40 K=25,49,2
      IF(K.GE.43)IDDT=4
      CALL GLRSET(IH,IV,1028,IBUF,NBYTE,MBYTE)
      CALL GLAN(IBCD(K),13,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(K),IDDT,K)
      IF(ICNT.LT.3.OR.K.EQ.47)GO TO 35
C  MOVE BEAM TO RIGHT TO DISPLAY NEXT COLUMN
      IH=IH+480
      ICNT=0
      IV=50
```

```
35     ICNT=ICNT+1
40     IV=IV-100
C   DISPLAY NEXT ORDER AND END PROGRAM BUTTONS
       IV=-550
       DO 50 KK=51,53,2
       CALL GURSET(-120,IV,102B,IBUF,NBYTE,MBYTE)
       CALL GUAN(IBCD(KK),11,IBUF,NBYTE,MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,IER(KK),8,KK)
50     IV=IV-200
C   CALL FOR BUTTON ID BLOCKS FOR ALL FOUR HEADINGS
       DO 120 II=1,4
55     CALL GIBUT(0,NCON,IDT,IDC)
C   WHEN BUTTON IS FOR A STRING OF DRINKS, BRANCH
       IF(IDC.EQ.23)GO TO 60
       IF(IDC.GE.51)GO TO 145
C   CALL FOR SINGLE PICKS
       CALL GIFID(NCON,IDTT,IDCC)
C   CHECK FOR ERRORS
       IF(IDCC.EQ.0)GO TO 200
       ICHK=2*II+15
       IF(ICHK.NE.IDC)JCK=1
       IF(ICHK.NE.IDC)GO TO 200
C   CHANGE THE BUTTON TO AN IGNORE ITEM
       CALL GIMOVE(-0,-0,-0,IER(IDC),1)
       GO TO 110
60     CONTINUE
C   CALL FOR STRING PICKS
C   SET ARRAYS TO ZERO
       DO 70 I=1,4
       IDTX(I)=0
70     IDCX(I)=0
       DO 80 LM=1,4
       NC=1
80     CALL GIFSID(NCON,NC,IDTX(LM),IDCX(LM))
C   CHECK FOR ERRORS
       IF(IDCX(1).EQ.0.AND.IDCX(2).EQ.0.AND.IDCX(3).EQ.0.AND.
      1IDCX(4).EQ.0)      GO TO 200
       ICHK=2*II+15
       IF(ICHK.NE.IDC)JCK=1
       IF(ICHK.NE.IDC)GO TO 200
       CALL GIMOVE(-0,-0,-0,IER(IDC),1)
C   ERASE UNPICKED DRINKS
       IIJ=0
       DO 100 J=43,49,2
       DO 90 I=1,4
       IF(J.EQ.IDCX(I))IIJ=J
       IF(IIJ.GT.0)GO TO 100
       IF(I.EQ.4.AND.IIJ.EQ.0)CALL GIERAS(IER(J))
90     CONTINUE
100    IIJ=0
       GO TO 130
C   ERROR PROCESSOR
200    IF(JCK.EQ.0)GO TO 210
       CALL GURSET(-500,1000,102B,IBUF,NBYTE,MBYTE)
       CALL GUAN(44HCHOOSE ITEMS FROM LEFT TO RIGHT--START AGAIN,44,IE
     1NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(54))
      GO TO 140
210   CALL GLRSET(-500,1000,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(40HPICK ONE ITEM BEFORE PICKING THE HEADING,40,IBUF,
     1NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IER(55))
      GO TO 55
C  ERASE ALL BUT PICKED ITEMS UNDER BUTTON
110   IF(IDC.EQ.17)IJ=25
      IF(IDC.EQ.19)IJ=31
      IF(IDC.EQ.21)IJ=37
C  ERASE ITEMS UNDER APPETIZER OR ENTREE OR DESSERT
      DO 120 JJ=1,5,2
      IK=JJ+IJ-1
      IF(IK.EQ.IDCC)GO TO 120
      CALL GIERAS(IER(IK))
120   CONTINUE
130   CONTINUE
C  ERASE REST OF CONSOLE AFTER NEXT ORDER BUTTON PICK
C  END JOB AFTER BUTTON PICK OF END JOB AND DISPLAY MESSAGE
140   CALL GICLR(NCON)
      CALL GIBUT(0,NCON,IDT,IDC)
      IF(IDC.LT.51)GO TO 140
145   IF(IDC.EQ.53)GO TO 170
C  ERASE SCREEN FOR NEXT ORDER
      DO 150 J=1,13,4
150   CALL GIERAS(IER(J))
      DO 160 I=17,47,6
160   CALL GIERAS(IER(I),IER(I+2),IER(I+4))
      CALL GIERAS(IER(53),IER(54),IER(55))
      GO TO 500
170   CALL GLRSET(-145,1500,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(13HPROGRAM ENDED,13,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IEND)
      CALL GICNRL(NCON)
      STOP
      END
```

STAR OF DAVID

```
JSTAR,P17,T10000,CM40000,TP1.
FTN.
LGO.
AEFILE.
SCR.
RELEASE,XMPL.
EXIT.
RELEASE,XMPL.
000000000000000000000
      OVERLAY(SCR,0,0)
      PROGRAM CREATE(INPUT,OUTPUT)
      CALL AEXEC
      END
      OVERLAY(1,0)
      PROGRAM LITTLE
      DIMENSION IBUF(64),IBCD(13),IBB(13)
      DATA IBCD(1)/40HSUPPOSED TO BE A SQUARE INSIDE A STAR    /
      DATA(IBCD(I),I=5,11,2)/20HMOVE SQUARE LEFT    ,
     120HMOVE SQUARE UP      ,20HMOVE SQUARE DOWN    ,
     120HMOVE SQUARE RIGHT   /
      DATA IBCD(13)/10HWRONG IDDT/
      DATA(IBB(K),K=1,9,2)/20HRELEASE             ,20HCHECK
     1    ,20HRESTART             ,20HVERY GOOD           ,
     120HTRY AGAIN           /
      NCON=9
C   CONNECT CONSOLE
      CALL GICNJB(NCON)
C   SET ITEM MASKS
      DO 10 I=2,5
      IDDTS=2**(I-1)
      IMASK=IDDTS
10    CALL GIMASK(NCON,-0,IDDTS,IMASK)
C     SET DISPLAY CONSTANTS
      ICODE=103B
      ISTYLE=7777B
      MBYTE=310
      NBYTE=0
C     START DRAWING SQUARE
      CALL GURSET(-1400,0,3B,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(-1400,0,ISTYLE,IBUF,NBYTE,MBYTE)
      CALL GUSEG(-1200,200,0)
      CALL GUSEG(-1200,-200,1)
      CALL GUSEG(-1600,-200,1)
      CALL GUSEG(-1600,200,1)
      CALL GUSEG(-1200,200,1)
C     DISPLAY SQUARE AS A SINGLE PICK ITEM
      CALL GIDISP(NCON,IBUF,NBYTE,IDRSV,2,1)
C     DRAW STAR OF DAVID (TWO TRIANGLES)
      CALL GURSET(-500,300,ICODE,IBUF,NBYTE,MBYTE)
      CALL GUSEGI(-500,300,ISTYLE,IBUF,NBYTE,MBYTE)
      CALL GUSEG(500,300,1)
      CALL GUSEG(0,-600,1)
      CALL GUSEG(-500,300,1)
      CALL GUSEG(-500,-300,0)
C     DRAW SECOND TRIANGLE
      CALL GUSEG(500,-300,1)
      CALL GUSEG(0,600,1)
      CALL GUSEG(-500,-300,1)
C    DISPLAY STAR AS STRING PICK WITH MARKER MASK SET
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,24B,2)
C  LABEL THE FIGURES
      CALL GURSET(-744,1400,3B,IBUF,NBYTE,MBYTE)
      CALL GUAN(62HMOVE THE SQUARE INTO THE STAR AND CHECK IF IT IS IN T
     1HE CENTER,62,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,1,3)
C     MAKE FOUR BUTTON CHOICES
      DO 20 J=5,11,2
      IF(J.EQ.5)IH=-1200
      IF(J.EQ.7)IH=-600
      IF(J.EQ.9)IH=0
      IF(J.EQ.11)IH=600
      CALL GURSET(IH,-900,ICODE,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBCD(J),20,IBUF,NBYTE,MBYTE)
20    CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,8,J)
      DO 52 N=13,21,2
      IDDT=8
      IV=-1200
      IF(N.EQ.13)IH=-105
      IF(N.EQ.13)IV=1800
      IF(N.EQ.15)IH=-96
      IF(N.EQ.17)IH=96
      IF(N.LT.19)GO TO 26
      IH=-175
      IV=1200
      ICODE=1B
      IDDT=1
26    M=N-12
      CALL GURSET(IH, IV   ,ICODE,IBUF,NBYTE,MBYTE)
      CALL GUAN(IBB(M),9,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,IDDT,N)
      M=0
      IH=-1400
      IV=0
      IF(N.EQ.19)IGOOD=IDDAD
      IF(N.EQ.21)ITRY=IDDAD
52    CONTINUE
      CALL GITCON(NCON,IH,IV)
      CALL GITIMV(NCON,IDRSV)
36    CALL GIBUT(0,NCON,IDDT,IDDC)
      IF(IDDC.LT.19.AND.IDDC.GT.3)GO TO 43
      CALL GIABRT(NCON,10HWRONG PICK,10)
      STOP
43    L=(IDDC-3 )/2
      CALL GITCOF(NCON,IH,IV)
      GO TO(44,45,46,47,48,49,60)L
44    IH=IH-50
      GO TO 50
45    IV=IV+50
      GO TO 50
46    IV=IV-50
      GO TO 50
47    IH=IH+50
50    CALL GIMOVE(IH,IV,103B,IDRSV)
      CALL GITCON(NCON,IH,IV)
      IF(M) 55,36,55
```

```
55     CALL GIMOVE(-2040,2040,1B,IGOOD)
       CALL GIMOVE(-2040,-2040,1B,ITRY)
       M=0
       GO TO 36
49     M=1
       CALL GITCON(NCON,IH,IV)
       IF(IABS(IH).LT.16.AND.IABS(IV).LT.16)GO TO 62
       CALL GIMOVE(-108,1700,7B,ITRY)
       CALL GIMOVE(-2040,-2040,1B,IGOOD)
       GO TO 36
62     CALL GIMOVE(-108,1200,7B,IGOOD)
       CALL GIMOVE(-2040,-2040,1B,ITRY)
       GO TO 36
60     IH=-1400
       IV=0
       GO TO 50
48     CALL GIABRT(NCON,12HJOB RELEASED,12)
       STOP
       END
000000000000000000000
XMPL
SCR
000000000000000000000
XMPL
000000000000000000000
000000000000000000000
```

SCISSORED ARC

```
       OVERLAY(ABC,0,0)
       PROGRAM CREATE(INPUT,OUTPUT)
       CALL AEXEC
       END


       OVERLAY(1,0)
       PROGRAM SCISR
       DIMENSION IBUF(64),IH1(5),IV1(5),IH2(5),IV2(5),JH1(5),JV1(5),
      1JH2(5),JV2(5),IBCD(3),INST(8),IAXS(2)
       DATA(INST(I),I=1,5,4)/40HPICK 5 CHARACTERS AND $ AS X AXIS LABEL ,
      1                      40HTHEN 5 CHARACTERS AND $ AS Y AXIS LABEL /
       MBYTE=310
       NBYTE=0
       NCON=9
600    CALL GICNJB(NCON)
C      SET MASKS
       DO 700 I=1,5
       N=2**(I-1)
700    CALL GIMASK(NCON,0,N,N)
C      ATTACH CONSOLE
C      DISPLAY REPEAT BUTTON AND SQUARE PLOT FRAME AND END JOB BUTTON
       CALL GURSET(-1200,-1200,102B,IBUF,NBYTE,MBYTE)
       CALL GUAN(6HREPEAT,6,IBUF,NBYTE,MBYTE)
       CALL GIDISP(NCON,IBUF,NBYTE,IRPT,30B,44)
       CALL GURSET(1000,-1000,2,IBUF,NBYTE,MBYTE)
       CALL GUSEGS(1000,-1000,-1000,-1000,1,7777B,IBUF,NBYTE,MBYTE)
       CALL GUSEG(-1000,1000,1)
       CALL GUSEG(1000,1000,1)
       CALL GUSEG(1000,-1000,1)
      CALL GIDISP(NCON,IBUF,NBYTE,IFRAM)
      CALL GLRSET(-1200,-1300,102B,IBUF,NBYTE,MBYTE)
      CALL GLAN(7HEND JOB,7,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,JBEND,30B,33)
C     DISPLAY SCISSORED ARC AND SCISSORED LINE
      CALL GLARC(0,0,1000,1000,1080.,-1080.,.1600.,0.,0.,-1600.,KSHOW,
     1IHC,IVC,IH1,IV1,IH2,IV2)
      IF(KSHOW,EQ.0)GOTO 800
      CALL GLRSET(IH1,IV1,2,IBUF,NBYTE,MBYTE)
      CALL GLARCG(KSHOW,IHC,IVC,IH1,IV1,IH2,IV2,7777B,
     1IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IARC)
800   CALL GLLINE(0,0,1000,1000,-800.,16.,1200.,-24.,KSHOW,JH1,JV1,
     1JH2,JV2)
      IF(KSHOW,EQ,0)GO TO 900
      CALL GLRSET(JH1,JV1,2,IBUF,NBYTE,MBYTE)
      CALL GLSEGS(JH1,JV1,JH2,JV2,1,7777B,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,LIN)
C     DISPLAY A FONT WITH $ AS EOM CHARACTER
900   CALL GFONTA(NCON,0,1500,IA,IB)
      CALL GIEOM(NCON,1R$,8,11)
C     DISPLAY INSTRUCTIONS
      CALL GLRSET(-650,-1400,3,IBUF,NBYTE,MBYTE)
      CALL GLAN(INST(1),40,IBUF,NBYTE,MBYTE)
      CALL GLRSET(-650,-1500,103B,IBUF,NBYTE,MBYTE)
      CALL GLAN(INST(5),40,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,INFOR)
C     ACCEPT FONT INPUT ONE CHARACTER AT A TIME
      K=0
1000  NC=6
      CALL GICLR(NCON)
      CALL GIANS(NCON,6,1400,1200)
1100  CALL GIBUT(0,NCON,IDDT,IDDC)
      IF(IDDC.GE,33)GO TO 1500
      IF(IDDC.NE,11)GO TO 1100
      CALL GIANE(NCON,NC,IBCD)
C     TEST FOR AT LEAST ONE CHARACTER
      IF(NC,EQ,1)GO TO 1000
      NC=NC-1
      IF(K,NE,0)GO TO 1200
C     DISPLAY AXIS LABELS----X THEN Y
      CALL GLRSET(-48,-1200,102B,IBUF,NBYTE,MBYTE)
      K=K+1
      GO TO 1300
1200  CALL GLRSET(-1150,0,2,IBUF,NBYTE,MBYTE)
      K=K+1
1300  CALL GLAN(IBCD,NC,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,IAXS(K))
      IF(K,EQ,1)GO TO 1000
C     WAIT FOR REPEAT BUTTON TO BE PICKED
1400  CALL GICLR(NCON)
      CALL GIBUT(0,NCON,IDDT,IDDC)
1500  IF(IDDC,NE,44,OR,IDDC,NE,33)GO TO 1400
C     CLEAR CONSOLE FOR A REPEAT RUN OR END JOB
      IF(IDDC,EQ,44)GO TO 600
      CALL GLRSET(1100,-1300,2,IBUF,NBYTE,MBYTE)
      CALL GLAN(9HJOB ENDED,9,IBUF,NBYTE,MBYTE)
      CALL GIDISP(NCON,IBUF,NBYTE,MSG)
      CALL GICNRL(NCON)
      STOP
      END
```

SAMPLE DATA HANDLER FILE DUMP B

This is a data handler dump from the circuit design program which begins on page A-24. As the following listing shows, each dump produced by a call to DMDMP is preceded by a line stating the relative number of the dumped block (decimal) and the amount of empty space within it (octal). Each bead in a block is preceded by a line stating its relative index number within the block (decimal). Continuation beads are marked and pointers to them are given after the printout of each file block. Word numbers within a bead are octal numbers.

Refer to page A-26 for a description of the information packed in each bead. A description of the first four beads of the dump is intended to illustrate reading of data handler dumps.

Bead 1 describes a node with one lead; the value 43000 points to the next node, and the pointer to the right element is 42000. The coordinates of the center of the circular node are $(-1440_8, 1440_8)$.

Bead 2 describes a horizontal line short from the node at 41000 into the node at 43000.

Bead 3 describes a node with three leads and the next node is at 45000. The left element is at 42000, the right is at 44000, and the down element is at 54000. The center coordinates of the node are $(-570_8, 1440_8)$.

Bead 4 describes a horizontal resistor from the node at 43000 into the node at 45000.

```
DUMP OF DATA HANDLER FILE -  STAN

BLOCK 0001 EMPTY SPACE 0006

BEAD  001

000001   00000 00000 00000 00007        00000 00000 00000 00001        00000 00000 00000 43000        00000 00000 00022 00022
000005   00000 00000 00000 00000  000007→00000 00000 00000 42000        00000 00000 00000 00000        77777 77777 77777 76337
000012   00000 00000 00000 01440

BEAD  002

000001   00000 00000 00000 00005        00000 00000 00000 41000        00000 00000 00000 43000        00000 00000 00022 00023
000005   00000 00000 00000 00000        00000 00000 00000 00000

BEAD  003

000001   00000 00000 00000 00007        00000 00000 00000 00003        00000 00000 00000 45000        00000 00000 00022 00024
000005   00000 00000 00000 00000        00000 00000 00000 42000        00000 00000 00000 44000        00000 00000 00000 54000
000011   77777 77777 77777 77207        00000 00000 00000 01440

BEAD  004

000001   00000 00000 00000 00001        00000 00000 00000 43000        00000 00000 00000 45000        00000 00000 00022 00025
000005   17317 64000 00000 00000        00000 00000 00022 00053

BEAD  005

000001   00000 00000 00000 00007        00000 00000 00000 00003        00000 00000 00000 47000        00000 00000 00022 00026
000005   00000 00000 00000 00000        00000 00000 00000 44000        00000 00000 00000 46000        00000 00000 00000 56000
000011   00000 00000 00000 00060        00000 00000 00000 01440

BEAD  006

000001   00000 00000 00000 00001        00000 00000 00000 45000        00000 00000 00000 47000        00000 00000 00022 00027
000005   17317 64000 00000 00000        00000 00000 00022 00057

BEAD  007

000001   00000 00000 00000 00007        00000 00000 00000 00003        00000 00000 00000 51000        00000 00000 00022 00030
000005   00000 00000 00000 00000        00000 00000 00000 46000        00000 00000 00000 50000        00000 00000 00001 02000
000011   00000 00000 00000 00730        00000 00000 00000 01440

BEAD  008

000001   00000 00000 00000 00005        00000 00000 00000 47000        00000 00000 00000 51000        00000 00000 00022 00031
000005   00000 00000 00000 00000        00000 00000 00000 00000

BEAD  009

000001   00000 00000 00000 00007        00000 00000 00000 00001        00000 00000 00000 53000        00000 00000 00022 00032
000005   00000 00000 00000 00000        00000 00000 00000 50000        00000 00000 00000 00000
000011   00000 00000 00000 01600        00000 00000 00000 01440

BEAD  010 RELEASED

BEAD  011

000001   00000 00000 00000 00007        00000 00000 00000 00001        00000 00000 00000 55000        00000 00000 00022 00034
000005   00000 00000 00000 00000  000007→00000 00000 00001 04000        00000 00000 00000 00000        77777 77777 77777 76337
000012   00000 00000 00000 00570
```

```
BEAD  012

000001   00000 00000 00000 00004     00000 00000 00000 43000     00000 00000 00000 55000     00000 00000 00022 00035
000005   17174 00000 00000 00000     00000 00000 00022 00063

BEAD  013

000001   00000 00000 00000 00007     00000 00000 00000 00003     00000 00000 00001 01000     00000 00000 00022 00036
000005   00000 00000 00000 54000     00000 00000 00001 04000     00000 00000 00001 05000     00000 00000 00000 00000
000011   77777 77777 77777 77207     00000 00000 00000 00570

BEAD  014

000001   00000 00000 00000 00004     00000 00000 00000 45000     00000 00000 00001 01000     00000 00000 00022 00037
000005   17174 00000 00000 00000     00000 00000 00022 00067

BLOCK 0002 EMPTY SPACE 0072

BEAD  001

000001   00000 00000 00000 00007     00000 00000 00000 00003     00000 00000 00001 03000     00000 00000 00022 00040
000005   00000 00000 00000 56000     00000 00000 00001 05000     00000 00000 00001 06000     00000 00000 00000 00000
000011   00000 00000 00000 00060     00000 00000 00000 00570

BEAD  002

000001   00000 00000 00000 00004     00000 00000 00000 47000     00000 00000 00001 03000     00000 00000 00022 00041
000005   17204 00000 00000 00000     00000 00000 00022 00073

BEAD  003

000001   00000 00000 00000 00007     00000 00000 00000 00003     00000 00000 00001 10000     00000 00000 00022 00042
000005   00000 00000 00001 02000     00000 00000 00001 06000     00000 00000 00001 07000     00000 00000 00000 00000
000011   00000 00000 00000 00730     00000 00000 00000 00570

BEAD  004

000001   00000 00000 00000 00005     00000 00000 00000 53000     00000 00000 00000 55000     00000 00000 00022 00043
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  005

000001   00000 00000 00000 00005     00000 00000 00000 55000     00000 00000 00001 01000     00000 00000 00022 00044
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  006

000001   00000 00000 00000 00005     00000 00000 00001 01000     00000 00000 00001 03000     00000 00000 00022 00045
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  007

000001   00000 00000 00000 00005     00000 00000 00001 03000     00000 00000 00001 10000     00000 00000 00022 00046
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  008

000001   00000 00000 00000 00007     00000 00000 00000 00001     00000 00000 00000 00000     00000 00000 00022 00047
000005   00000 00000 00000 00000     00000 00000 00001 07000     00000 00000 00000 00000
000011   00000 00000 00000 01600     00000 00000 00000 00570
```

```
DUMP OF DATA HANDLER FILE -  STAN

BLOCK 0001 EMPTY SPACE 0014

BEAD  001

000001   00000 00000 00000 00007          00000 00000 00000 00001          00000 00000 00000 43000          00000 00000 00022 00022
000005   00000 00000 00000 00000   000007↱00000 00000 00000 42000          00000 00000 00000 00000          77777 77777 77777 76337
000012   00000 00000 00000 01440

BEAD  002

000001   00000 00000 00000 00005          00000 00000 00000 41000          00000 00000 00000 43000          00000 00000 00022 00023
000005   00000 00000 00000 00000          00000 00000 00000 00000

BEAD  003

000001   00000 00000 00000 00007          00000 00000 00000 00003          00000 00000 00000 45000          00000 00000 00022 00024
000005   00000 00000 00000 00000          00000 00000 00000 42000          00000 00000 00000 44000          00000 00000 00000 54000
000011   77777 77777 77777 77207          00000 00000 00000 01440

BEAD  004

000001   00000 00000 00000 00001          00000 00000 00000 43000          00000 00000 00000 45000          00000 00000 00022 00025
000005   17317 64000 00000 00000          00000 00000 00022 00053

BEAD  005

000001   00000 00000 00000 00007          00000 00000 00000 00002          00000 00000 00000 47000          00000 00000 00022 00026
000005   00000 00000 00000 00000          00000 00000 00000 44000          00000 00000 00000 46000          00000 00000 00000 00000
000011   00000 00000 00000 00060          00000 00000 00000 01440

BEAD  006

000001   00000 00000 00000 00001          00000 00000 00000 45000          00000 00000 00000 47000          00000 00000 00022 00027
000005   17317 64000 00000 00000          00000 00000 00022 00057

BEAD  007

000001   00000 00000 00000 00007          00000 00000 00000 00003          00000 00000 00000 51000          00000 00000 00022 00030
000005   00000 00000 00000 00000          00000 00000 00000 46000          00000 00000 00000 50000          00000 00000 00001 02000
000011   00000 00000 00000 00730          00000 00000 00000 01440

BEAD  008

000001   00000 00000 00000 00005          00000 00000 00000 47000          00000 00000 00000 51000          00000 00000 00022 00031
000005   00000 00000 00000 00000          00000 00000 00000 00000

BEAD  009

000001   00000 00000 00000 00007          00000 00000 00000 00001          00000 00000 00000 53000          00000 00000 00022 00032
000005   00000 00000 00000 00000          00000 00000 00000 50000          00000 00000 00000 00000
000011   00000 00000 00000 01600          00000 00000 00000 01440

BEAD  010 RELEASED

BEAD  011

000001   00000 00000 00000 00007          00000 00000 00000 00001          00000 00000 00000 55000          00000 00000 00022 00034
000005   00000 00000 00000 00000   000007↱00000 00000 00001 04000          00000 00000 00000 00000          77777 77777 77777 76337
000012   00000 00000 00000 00570
```

```
BEAD  012

000001   00000 00000 00000 00004       00000 00000 00000 43000       00000 00000 00000 55000       00000 00000 00022 00035
000005   17174 00000 00000 00000       00000 00000 00022 00063

BEAD  013

000001   00000 00000 00000 00007       00000 00000 00000 00002       00000 00000 00001 03000       00000 00000 00022 00036
000005   00000 00000 00000 54000       00000 00000 00001 04000       00000 00000 00000 00000
000011   77777 77777 77777 77207       00000 00000 00000 00570

BEAD  014 RELEASED

BLOCK 0002 EMPTY SPACE 0120

BEAD  001 RELEASED

BEAD  002

000001   00000 00000 00000 00004       00000 00000 00000 47000       00000 00000 00001 03000       00000 00000 00022 00041
000005   17204 00000 00000 00000       00000 00000 00022 00073

BEAD  003

000001   00000 00000 00000 00007       00000 00000 00000 00002       00000 00000 00001 10000       00000 00000 00022 00042
000005   00000 00000 00001 02000       00000 00000 00000 00000       00000 00000 00001 07000       00000 00000 00000 00000
000011   00000 00000 00000 00730       00000 00000 00000 00570

BEAD  004

000001   00000 00000 00000 00005       00000 00000 00000 53000       00000 00000 00000 55000       00000 00000 00022 00043
000005   00000 00000 00000 00000       00000 00000 00000 00000

BEAD  005 RELEASED

BEAD  006 RELEASED

BEAD  007

000001   00000 00000 00000 00005       00000 00000 00001 03000       00000 00000 00001 13000       00000 00000 00022 00046
000005   00000 00000 00000 00000       00000 00000 00000 00000

BEAD  008

000001   00000 00000 00000 00007       00000 00000 00000 00031       00000 00000 00000 00000       00000 00000 00022 00047
000005   00000 00000 00000 00000       00000 00000 00001 07000       00000 00000 00000 00000
000011   00000 00000 00000 01600       00000 00000 00000 00570
```

```
DUMP OF DATA HANDLER FILE -  STAN

BLOCK 0001 EMPTY SPACE 0022

BEAD  001

000001   00000 00000 00000 00007         00000 00000 00000 00001        00000 00000 00000 43000        00000 00000 00022 00022
000005   00000 00000 00000 00000  000007+00000 00000 00000 42000        00000 00000 00000 00000        77777 77777 77777 76337
000012   00000 00000 00000 01440

BEAD  002

000001   00000 00000 00000 00005         00000 00000 00000 41000        00000 00000 00000 43000        00000 00000 00022 00023
000005   00000 00000 00000 00000         00000 00000 00000 00000

BEAD  003

000001   00000 00000 00000 00007         00000 00000 00000 00003        00000 00000 00000 45000        00000 00000 00022 00024
000005   00000 00000 00000 00000         00000 00000 00000 42000        00000 00000 00000 44000        00000 00000 00000 54000
000011   77777 77777 77777 77207         00000 00000 00000 01440

BEAD  004

000001   00000 00000 00000 00001         00000 00000 00000 43000        00000 00000 00000 45000        00000 00000 00022 00025
000005   17317 64000 00000 00000         00000 00000 00022 00053

BEAD  005

000001   00000 00000 00000 00007         00000 00000 00000 00002        00000 00000 00000 47000        00000 00000 00022 00026
000005   00000 00000 00000 00000         00000 00000 00000 44000        00000 00000 00000 46000        00000 00000 00000 00000
000011   00000 00000 00000 00060         00000 00000 00000 01440

BEAD  006

000001   00000 00000 00000 00001         00000 00000 00000 45000        00000 00000 00000 47000        00000 00000 00022 00027
000005   17317 64000 00000 00000         00000 00000 00022 00057

BEAD  007

000001   00000 00000 00000 00007         00000 00000 00000 00003        00000 00000 00000 51000        00000 00000 00022 00030
000005   00000 00000 00000 00000         00000 00000 00000 46000        00000 00000 00000 50000        00000 00000 00001 02000
000011   00000 00000 00000 00730         00000 00000 00000 01440

BEAD  008

000001   00000 00000 00000 00005         00000 00000 00000 47000        00000 00000 00000 51000        00000 00000 00022 00031
000005   00000 00000 00000 00000         00000 00000 00000 00000

BEAD  009

000001   00000 00000 00000 00007         00000 00000 00000 00001        00000 00000 00000 55000        00000 00000 00022 00032
000005   00000 00000 00000 00000         00000 00000 00000 50000        00000 00000 00000 00000
000011   00000 00000 00000 01600         00000 00000 00000 01440

BEAD  010
CONTINUATION BEAD

000001   00000 00000 00001 41000         00000 00000 00022 00107        00000 00000 00000 00000        00000 00000 00000 00000

BEAD  011 RELEASED
BEAD  012

000001   00000 00000 00000 00004          00000 00000 00000 43000          00000 00000 00000 55000          00000 00000 00022 00035
000005   17174 00000 00000 00000          00000 00000 00022 00063

BEAD  013

000001   00000 00000 00000 00007          00000 00000 00000 00002          00000 00000 00001 03000          00000 00000 00022 00036
000005   00000 00000 00000 54000          00000 00000 00000 00000    000010+00000 00000 00001 05000          77777 77777 77777 77207
000012   00000 00000 00000 00570

BEAD  014 RELEASED

BLOCK 0002 EMPTY SPACE 0006

BEAD  001

000001   00000 00000 00000 00007          00000 00000 00000 00003          00000 00000 00001 12000          00000 00000 00022 00077
000005   00000 00000 00001 05000          00000 00000 00001 13000          00000 00000 00001 14000          00000 00000 00000 00000
000011   77777 77777 77777 77207          77777 77777 77777 77717

BEAD  002

000001   00000 00000 00000 00002          00000 00000 00000 47000          00000 00000 00001 03000          00000 00000 00022 00100
000005   17337 64000 00000 00000          00000 00000 00022 00120

BEAD  003

000001   00000 00000 00000 00007          00000 00000 00000 00002          00000 00000 00001 10000          00000 00000 00022 00042
000005   00000 00000 00001 02000          00000 00000 00000 00000          00000 00000 00001 07000          00000 00000 00001 11000
000011   00000 00000 00000 00730          00000 00000 00000 00570

BEAD  004 RELEASED

BEAD  005

000001   00000 00000 00000 00002          00000 00000 00000 55000          00000 00000 00001 01000          00000 00000 00022 00076
000005   17307 64000 00000 00000          00000 00000 00022 00114

BEAD  006

000001   00000 00000 00000 00007          00000 00000 00000 00001          00000 00000 00001 01000          00000 00000 00022 00075
000005   00000 00000 00000 00000    000007+00000 00000 00001 13000          00000 00000 00000 00000          77777 77777 77777 76337
000012   77777 77777 77777 77717

BEAD  007

000001   00000 00000 00000 00005          00000 00000 00001 03000          00000 00000 00001 10000          00000 00000 00022 00046
000005   00000 00000 00000 00000          00000 00000 00000 00000

BEAD  008

000001   00000 00000 00000 00007          00000 00000 00000 00001          00000 00000 00001 06000          00000 00000 00022 00047
000005   00000 00000 00000 00000          00000 00000 00001 07000          00000 00000 00000 00000
000011   00000 00000 00000 01600          00000 00000 00000 00570

BEAD  009

000001   00000 00000 00000 00002          00000 00000 00001 03000          00000 00000 00001 12000          00000 00000 00022 00101
000005   17354 70400 00000 00000          00000 00000 00022 00124

BEAD  010
```

```
000001   00000 00000 00000 00007     00000 00000 00000 00003     00000 00000 00001 15000     00000 00000 00022 00102
000005   00000 00000 00001 11000     00000 00000 00001 16000     00000 00000 00001 17000     00000 00000 00000 00000
000011   00000 00000 00000 00730     77777 77777 77777 77717

BEAD  011

000001   00000 00000 00000 00005     00000 00000 00001 06000     00000 00000 00001 01000     00000 00000 00022 00103
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  012

000001   00000 00000 00000 00005     00000 00000 00001 01000     00000 00000 00001 15000     00000 00000 00022 00104
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  013

000001   00000 00000 00000 00007     00000 00000 00000 00002     00000 00000 00001 41000     00000 00000 00022 00105
000005   00000 00000 00000 00000     00000 00000 00001 14000     00000 00000 00001 16000     00000 00000 00000 00000
000011   00000 00000 00000 00060     77777 77777 77777 77717

BEAD  014

000001   00000 00000 00000 00005     00000 00000 00001 15000     00000 00000 00001 12000     00000 00000 00022 00106
000005   00000 00000 00000 00000     00000 00000 00000 00000

BEAD  015

000001   00000 00000 00000 00005     00000 00000 00001 12000

                  CONTINUE AS BLOCK 0001 BEAD  010

BLOCK 0003 EMPTY SPACE 0163

BEAD  001

000001   00000 00000 00000 00007     00000 00000 00000 00001     00000 00000 00000 00000     00000 00000 00022 00110
000005   00000 00000 00000 00000     00000 00000 00001 17000     00000 00000 00000 00000
000011   00000 00000 00000 01600     77777 77777 77777 77717
```

MISCELLANEOUS CODING HINTS C

ALPHANUMERIC INPUT GIANE (NCON, IALF, NC)

GIANE allows the calling routine to end alphanumeric input and transfer the alphanumeric buffer to IALF. It also transfers the number of characters returned in NC.

If a fixed number of characters are always returned (example, one digit), DECODE can be used to put the display code digit in an integer format on a fixed FORMAT statement:

```
      CALL GIANE (NCON,NC,IALF)
      DECODE (1,100,IALF) INT
100   FORMAT (I1)
```

INT - is an integer digit converted from the display code representation (of an integer digit).

If NC varies over a certain range (for example from 1 to 10), ENCODE can be used to construct a variable format to decode the integer.

```
      CALL GIANE(NCON,NC,IALF)
      IF (NC .LT. 1 .OR. NC. GT. 10)20, 10
 10   ENCODE( 5,100,SPEC)NC
      DECODE( NC,SPEC,IALF)INT
           ●
           ●
           ●
           ●
           ●
           ●
 20   ...
100   FORMAT( 2H(I,I2,1H))
```

After statement 10, location SPEC contains the correct format for decoding NC integer digits from IALF.

For example, if NC=4, the statement becomes

```
ENCODE (5,100,SPEC)4
DECODE(4,SPEC,IALF) INT
```

SPEC - contains $\boxed{\text{(I04)}}$

INT – contains the integer representation of the four display code digits from IALF.

ILLEGAL CHARACTERS

Since illegal characters in the buffers of ENCODE and DECODE result in fatal FORTRAN errors, the parameters should be checked before calling these routines.

For example, a display code 9 (44_8) will cause a fatal error if decoded on an O1 (octal) format. Fixed formats should not be used if variable results are expected.

ID WORD MANIPULATION

IDWA and IDWB are 24-bit quantities used by the task return to assemble a task name for load and execution.

AETSKR asks for the next button on the queue and assembles a left-justified display code task name.

Bits	59 - 36	IDWA	Bits 23 - 0
Bits	35 - 18	IDWB	Bits 23 - 6

If the calling routine is to request a task whose name is in IDWA and IDWB after it has retrieved the button from the queue, ENCODE and DECODE can be used to pack the contents of the two words into a left-justified display code name.

Example:

```
     IDWA      =      000000TASK
     IDWB      =      0000000ONE
        DECODE(10,1,IDWA)TEMP1
        DECODE(10,1,IDWB)TEMP2
        ENCODE(8,2,NAME)TEMP1,TEMP2
      1 FORMAT(6X,A4)
      2 FORMAT(2A4)
  TEMP1=TASK000000
  TEMP2=ONE 00
```

SUBROUTINE LINKAGE AMONG OVERLAYS

System and user subroutines which are frequently called by different overlays of a graphics job may be included in the (0, 0) overlay. Although the routines will stay in memory throughout execution and the length of the (0, 0) overlay will be increased, the decks need not be included with any primary or secondary overlay decks if they are compiled and loaded with (0, 0); the binary text will not appear on the task file.

The following example deck and outline of the GPSL loader's overlay load will demonstrate linkage of subroutines among overlays.

```
OVERLAY(A,0,0)
```
OVERLAY(l_1,l_2,CNNNNNN)

A	- is the name of GPSL loader save file
l_1	- is the primary overlay level
l_2	- is the secondary overlay level
NNNNNN	- is an octal digit. If the C parameter is present, the overlay will be set NNNNNN words from the beginning of blank COMMON. If no C parameter is present, the overlay will be placed at the end of blank COMMON. If no blank COMMON is present, the overlay will be placed at the end of the (0, 0) overlay text.

Example Deck:

```
                OVERLAY(A,0,0)
               ┌PROGRAM ZERO          ┐
ZERO          <│CALL MAIN             │
               └END                   │
               ┌SUBROUTINE USEFUL     │
USEFUL        <│RETURN                >DECK (0,0)
               └END                   │
               ┌SUBROUTINE NEEDED     │
NEEDED        <│RETURN                │
               └END                   ┘

                OVERLAY(1,0,C200)
               ┌PROGRAM ONE           ┐
               │CALL USEFUL           │
ONE           <│CALL SUB              │
               │CALL SUBB             >DECK (1,0)
               └END                   │
               ┌SUBROUTINE SUB        │
SUB           <│RETURN                │
               └END                   ┘

                OVERLAY(1,1)
               ┌PROGRAM ONEONE        ┐
ONEONE        <│CALL NEEDED           │
               │CALL SUBB             │
               └END                   >DECK (1,1)
               ┌SUBROUTINE SUBB       │
SUBB          <│RETURN                │
               └END                   ┘
```

```
           OVERLAY(2,0)
          (PROGRAM TWO          )
TWO       (CALL USEFUL          )
          (CALL SUBB            )
          (END                  )
                                 > DECK (2,0)
          (SUBROUTINE SUBB      )
SUBB      (RETURN               )
          (END                  )
```

The GPSL loader will read in all of the decks for OVERLAY(0, 0); in this case, the program ZERO and subroutines USEFUL and NEEDED. All linking will be done and loading completed by searching the library and filling in external references. If blank COMMON references are present, the largest blank COMMON reference will be placed immediately following OVERLAY(0, 0).

For the current example, assume that a 2000B word block of blank COMMON is placed immediately following OVERLAY(0, 0). Once the building of OVERLAY(0, 0) is complete, it is written on file A.

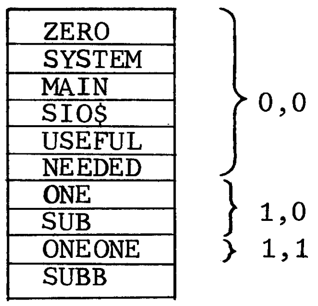
The next OVERLAY(1, 0) is read into the same area vacated by (0, 0). However, virtual address relocation is performed on the program text. In the current example, the base address for relocation of (1, 0) is the origin of blank COMMON plus 200B.

OVERLAY(1, 1) is processed in the same manner. The base address for relocation is the last word address plus one (LWA+1) of OVERLAY(1, 0). The base address for relocation of OVERLAY(2, 0) is the LWA+1 of the 2000-word blank COMMON block associated with (0, 0).

External references for OVERLAY(1, 0) are satisfied by searching the routines available in (0, 0), and the library.

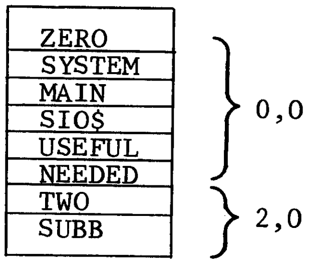
In the example, assuming that SUBB is not a library routine, the deck for SUBB is not present in OVERLAY(1, 0) and the deck is not present in OVERLAY(0, 0), SUBB will be an unsatisfied external in OVERLAY(1, 0).

The subroutines in memory after the task call of ONE and ONEONE would be:



SYSTEM and SIO$ are routines loaded to satisfy externals in program ZERO (or any FORTRAN program). References in ONE and ONEONE to entry points of these routines are linked to zero-zero. The reference to USEFUL in ONE and the reference to NEEDED in ONEONE are linked to their occurrence in (0, 0). The reference to SUB in (1, 1) is linked to its occurrence in (1, 0). Although SUBB occurs in (1, 1), it is an unsatisfied external in (1, 0), because the deck is encountered after the references for (1, 0) have been satisfied.

The routines in memory after a task call to TWO are:



There are no unsatisfied externals.

The (0, 0) OVERLAY remains in memory throughout execution. All routines included will be linked to any references in any other overlay.

External references in secondary overlays ((1, 1), (1, 2), etc.) will be linked to their occurrences in the primary overlay with the same primary overlay number.

INCLUDING SUBROUTINES IN THE (0,0) OVERLAY

USER ROUTINES

To include a user-written routine in the zero-zero overlay, the source deck must be compiled and loaded with the main program of that overlay. However, the deck need not be included in any other primary or secondary overlay which calls that routine.

SYSTEM LIBRARY ROUTINES

There are two simple ways to include system routines in the zero-zero overlay:

- FORTRAN callable routines may be included by placing a CALL name card in (0, 0). If the call statement appears after the CALL MAIN card, it will never be executed; however, the routine will be loaded with (0, 0) and all other CALL name references will be linked to the routine's occurrence in (0, 0).

- A COMPASS program in (0, 0) containing EXT name pseudo-ops will cause routine name to be loaded with (0, 0), as shown in the following example:

  Routine PSEUD compiled and loaded with (0, 0)

  ```
  IDENT    PSEUD
  EXT      GIDISP
  EXT      GIERAS
  END
  ```

  This routine will cause GIERAS and GIDISP to be loaded with the (0, 0) overlay.

PARAMETERS AND CALL SEQUENCES D

GENERAL

This appendix is a summary of IGS parameters and call sequences. The most common ID block parameters are:

IBEAM	±0 indicates light beam off ±1 indicates light beam on in accordance with bit pattern in ISTYLE
IBUF	Item description buffer parameter
ICODE	Governs light-pen sensitivity, blink, and brightness of a graphics display item. ICODE is usually expressed in octal (see Chapter 1).
IDDAD	Associative address assigned by the system to a display item
IDDC	ID code word assigned by the programmer $0 \leq \text{IDDC} \leq 2^8-1$
IDDT	ID-type code to specify how the queue handler will treat the button ID block.
IH	Horizontal axis coordinate
IMASK	Mask indicator code used by GIMASK where 1 = ignore 2 = single pick 4 = string pick 8 = button mask 16 = marker
IR	Code to control GIBUT return 0 = wait for button pick ID block to be queued 1 = return to calling task immediately
ISTYLE	Determines appearance of a line segment 0, -0, 7777B = solid segment 5252B = dashed segment 6666B = broken segment 7272B = center line segment style

IV	Vertical axis coordinate
MAD	Display buffer associative address of a macro returned by the call
MBYTE	Maximum number of bytes programmer allows to be placed in IBUF ≤ 322
NBYTE	Number of bytes currently in IBUF; returned as a result of the call
NC	Number of characters from IBCD to be packed by the call
NCON	Number of the graphics console referenced through the call

Undetermined parameters are required and a right parenthesis can terminate the parameter string any time after the required parameters.

In the listings given below, an underline (–) indicates that parameters are required.

CONSOLE CONTROL

	Ref. Page
GICNJB (NCON)	3-7
GICNRL (NCON)	5-9

DISPLAY ITEM OR MACRO CREATION

GIDISP (NCON,IBUF,NBYTE,IDDAD, IDDT,IDDC,IDWA,IDWB)	2-8
GIMAC (NCON,IBUF,NBYTE,MAD)	3-2

BYTE GENERATION FOR DISPLAY ITEM DESCRIPTIONS

GURSET (IH,IV,ICODE,IBUF,NBYTE,MBYTE)	2-5
GUSEGS (IH1,IV1,IH2,IV2,IBEAM,ISTYLE,IBUF,NBYTE,MBYTE)	2-6
GUSEGI (IH1,IV1,ISTYLE,IBUF,NBYTE,MBYTE)	2-7
GUSEG (IH,IV,IBEAM)	2-6
GUSEGA (IH,IV,IBEAM,N,ISTYLE,IBUF,NBYTE,MBYTE)	2-7
GUARCG (KSHOW,IHC,IVC,IH1,IV1,IH2,IV2,ISTYLE,IBUF, NBYTE,MBYTE)	2-10
GUAN (IBCD,NC,IBUF,NBYTE,MBYTE)	2-9
GUBYTE (IBYTE,L,IBUF,NBYTE,MBYTE)	A-8
GUMACG (MAD,L,IBUF,NBYTE,MBYTE)	3-3

DISPLAY EDITING

GIMOVE (IH,IV,ICODE,IDDAD,IDDT,IDDC,IDWA,IDWB)	3-4
GICOPY (IDDADI,NCON,IH,IV,ICODE,IDDAD, IDDT,IDDC, IDWA,IDWB)	3-6
GIERAS (IDDAD(1), IDDAD(2),....IDDAD(36))	3-6
GIMACE (MAD(1), MAD(2),...MAD(36))	3-8

SPECIAL ID BLOCK ASSIGNMENT

GIKYBD (NCON, IDDT,IDDC,IDWA,IDWB)	
GILPKY (NCON, IDDT,IDDC,IDWA,IDWB)	3-4
GIEOM (NCON, IBCD,IDDT,IDDC,IDWA,IDWB)	5-4
GIPBUT (NCON, IIDDT,IDDT,IDDC,IDWA,IDWB)	A-26

TRACKING CONTROL

GITCON (NCON,IH,IV)	3-3
GITCOF (NCON,IH,IV)	3-5
GITIMV (NCON,IDDAD)	3-3
GITMMV (NCON,MAD)	3-3

INPUT

CONTROL

GIMASK (NCON,IDDTC,IDDTS,IMASK)	3-9
GICLR (NCON)	3-4

PICK FETCHING

GIBUT (IR,NCON,IDDT,IDDC,IDWA,IDWB,IH,IV)	3-4
GIFID (NCON,IDDT,IDDC,IDWA,IDWB,IH,IV)	4-8
GIFSID (NCON,N,IDDT,IDDC,IDWA,IDWB,IH,IV)	4-8

ALPHANUMERIC

GIANS (NCON,NC,IH,IV)	5-5
GIANE (NCON,NC,IBUF)	5-6

VOLUNTARY JOB ABORT

GIABRT (NCON,IALF,NC)	3-8

GRAPHICS FUNCTION (6000 SERIES ONLY)

FRAME-SCISSORING

```
GULINE (IHCEN,IVCEN,IHCOR,IVCOR,H1,V1,H2,V2,KSHOW,          5-3
        IH1,IV1,IH2,IV2)
GUARC (IHCEN,IVCEN,IHCOR,IVCOR,HC,VC,H1,V1,H2,V2,           5-3
       KSHOW,IHC,IVC,IH1,IV1,IH2,IV2)
```

DISPLAY FONT GENERATION

```
GFONTA (NCON,IH,IV,IDADA,IDADB)                             5-4
GFONTN (NCON,IH,IV,IDADN)                                   5-6
```

HARDCOPY FILE CREATION

```
GIPLOT (NCON,IBUF,NBYTE,IDENT,ITYPE)
```

EXTRA PARAMETER IN CALL SEQUENCE

Placing an extra parameter in the calling sequence is just as fatal as omitting a required parameter. For example:

CORRECT

```
CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,IDDT,IDDC,-0)
CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,IDDT,IDDC)
```

WRONG

```
CALL GIDISP(NCON,IBUF,NBYTE,-0)
CALL GIDISP(NCON,IBUF,NBYTE,IDDAD,IDDT,IDDC,IDWA,IDWB,-0)
```

Parameter lists which are too short or too long will not be specifically diagnosed but will result in mode errors or nonsense stored in a data area or the entry address of an IGS system subroutine.

Care should be exercised in calling subroutines which return information. For example, IDDAD is the fourth parameter in the calling sequence to GIDISP. The display associative address will be returned to IDDAD's location.

WRONG

```
CALL GIDISP(NCON, IBUF, NBYTE, -0)
```

This causes the destruction of the contents of the location represented by the literal -0.

ANSWER KEY FOR REVIEW QUESTIONS

SECTION 2

1-	B	6-	B	11-	B, C, G	16-	B	21-	B
2-	B	7-	C	12-	A, C, F	17-	B		
3-	A	8-	C	13-	B, D, G	18-	B		
4-	C	9-	C	14-	B, C	19-	C		
5-	A	10-	A	15-	B	20-	B		

SECTION 3

1-	A	6-	B	11-	C	16-	A
2-	C	7-	C	12-	C		
3-	B	8-	C	13-	A		
4-	A	9-	B	14-	A		
5-	B	10-	B	15-	C		

SECTION 4

1-	A	6-	C	11-	B	16-	T
2-	C	7-	B	12-	T	17-	F
3-	C	8-	A	13-	F		
4-	C	9-	C	14-	T		
5-	C	10-	A	15-	T		

SECTION 5

1-	A	2-	C	3-	C	4-	B	5-	B

SOLUTIONS TO IGS CODING PROBLEMS F

Assume:

NCON	=	1
MBYTE	=	312

Masks are

1	=	ignore
2	=	single pick
4	=	string pick
8	=	button pick
16	=	marker mask

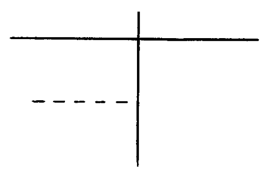
SECTION 2

1.
```
      CALL GURSET (1000, 500, 106B, IBUF, NBYTE, MBYTE)
      CALL GUARCG (1, 500, 500, 1000, 500, 1000, 500,
     1    7777B, IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 1)
```

2. a.
```
      CALL GURSET (-120, -12, 103B, IBUF, NBYTE,
     1     MBYTE
      CALL GUAN (10HJOHN SMITH, 10, IBUF, NBYTE,
     1     MBYTE)
      CALL GIDISP (NCON,IBUF, NBYTE, IDDAD)
```

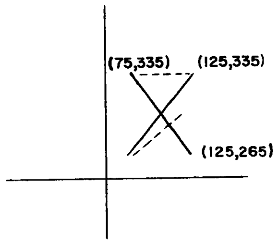
b.
```
      NAME = 10HJOHN SMITH
      CALL GURSET (-120, -12, 3, IBUF, NBYTE, MBYTE)
      CALL GUAN (NAME, 10, IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)
```

3. a.
```
      CALL GURSET (-1500, -1000, 103B, IBUF, NBYTE,
     1     MBYTE)
      CALL GUSEGS (-1500, -1000, 0, -1000, 1, 5252B,
     1     IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)
```

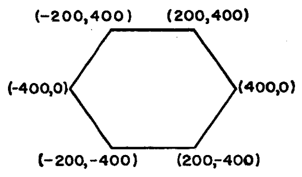


b.
```
      CALL GURSET (-1500, -1000, 103B, IBUF, NBYTE,
     1     MBYTE)
      CALL GUSEGS (0, 0,1500, 0, 1, 0, IBUF, NBYTE,
     1     MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)
```

```
4.     CALL GURSET (100, 100, 103B, ITRI, NBYTE, MBYTE)
       CALL GUSEGI (100, 100, 7777B, ITRI, NBYTE, MBYTE)
       CALL GUSEG (200, 0, 1)
       CALL GUSEG (0, 0, 1)
       CALL GUSEG (100, 100, 1)
       CALL GIDISP (NCON, ITRI, NBYTE, ITRNG)


5.     CALL GURSET (-100, 0, 103B, ISEMI, NBYTE, MBYTE)
       CALL GUSEGS (-100, 0, -300, 0, 1, 0, ISEMI, NBYTE,
      1     MBYTE)
       CALL GUARCG (1, -200, 0, -300, 0, -100, 0, 0, ISEMI,
      1     NBYTE, MBYTE)
       CALL GIDISP (NCON, ISEMI, NBYTE, IDDAD)


6.     DIMENSION IH1 (3), IH2 (3), IV1 (3), IV2 (3)
       CALL GURSET (-300, 0, 3, IBUF, NBYTE, MBYTE)
       IH1(1)=-300 $IH2(1)=-200 $IV1(1)=0 $IV2(1)=-100
       IH1(2)=-200 $IH2(2)=-100 $IV1(2)=-100 $IV2(2)=0
       IH1(3)=-100 $IH2(3)=-200 $IV1(3)=0 $IV2(3)=100
       CALL GUARCG (3, -200, 0, IH1, IV1, IH2, IV2, ISTYLE,
      1     IBUF, NBYTE, MBYTE)
       CALL GUSEG  (-200, 100, -200, 0, 1, 0, IBUF, NBYTE,
      1     MBYTE)
       CALL GUSEG  (-300, 0, 1)
       CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 1)


7.     a.     CALL GURSET (100, 300, 103B, IBUF, NBYTE,
             1     MBYTE)
              CALL GUSEGI (0, 0, 0, IBUF, NBYTE, MBYTE)
              CALL GUSEG (-25, 35, 0)
              CALL GUSEG (25, -35, 1)
              CALL GUSEG (-25, -35, 0)
              CALL GUSEG (25, 35, 1)
              CALL GUSEG (100, -12, 0)
            C DISPLAY COORDINATE LABEL
              ICORD = 10H(100, 300)
              CALL GUAN (ICORD, 10, IBUF, NBYTE, MBYTE)
              CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)

       b.     CALL GURSET (125, 265, 103B, IBUF, NBYTE,
             1     MBYTE)
              CALL GUSEGI (125, 265, 0, IBUF, NBYTE, MBYTE)
              CALL GUSEG (75, 335, 1)
              CALL GUSEG (125, 335, 0)
              CALL GUSEG (75, 265, 1)
            C MOVE BEAM TO WRITE COORDINATE LABEL
              CALL GUSEG (200, 300, 0)
              CALL GUAN (9H(100,300), 9, IBUF, NBYTE, MBYTE)
              CALL GIDISP (NCON, IBUF, NBYTE, IDDAD)
```

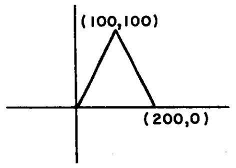


```
8.    CALL GURSET (-200, 400, 103B, IBUF, NBYTE, MBYTE)
      CALL GUSEGA (IH(1), IV(1), IBEAM(1), 6, 7777B, IBUF,
     1     NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IHEX)
```

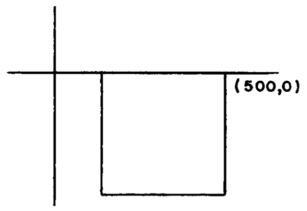


SECTION 3

```
1.    CALL GIMAC (NCON, ITRI, NBYTE, ITRNGL)
      CALL GURSET (100, 100, 103B, ITRI, NBYTE, MBYTE)
      CALL GUMACG (ITRNGL, 1, ITRI, NBYTE, MBYTE)
      CALL GIDISP (NCON, ITRI, NBYTE, IDDAD, 8)
```

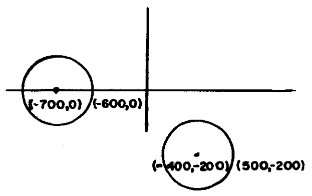


```
2.    a.     DO 10 I = 1,5
             N = 2 ** (I-1)
          10 CALL GIMASK (NCON, 0, N, N)

      b.     CALL GIMASK (NCON, 0, 1, 1)
             CALL GIMASK (NCON, 0, 2, 2)
             CALL GIMASK (NCON, 0, 4, 4)
             CALL GIMASK (NCON, 0, 8, 8)
             CALL GIMASK (NCON, 0, 16, 16)
```



```
3.    CALL GICOPY (SQR, NCON, 500, 0, 103B, NEWSQ, 2)


4.           CALL GITCON (NCON, 200, 300)
      1000   CALL GIBUT (0, NCON, IDDAD, IDDT, IDDC)
             IF(IDDC.NE.200) GO TO 1000
                       .
                       .
                       .
             CALL GITCOF (NCON, IHA, IVA)


5.    CALL GURSET (IHA, IVA, 103B, IBUF, NBYTE, MBYTE)
      CALL GUARCG (1, IHA -550, IVA -550, IHA, IVA, IHA,
     1     IVA, 0, IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 2)


6.    CALL GURSET (-1000, -1000, 102B, IBUF, NBYTE, MBYTE)
      CALL GUAN (5HCLEAR, 5, IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, ICLR, 8, 2, 400,
     1     4RCLER)


7. 11 CALL GIBUT (0, NCON, IDDT, IDDC)
      IF (IDDC. NE. 2) GO TO 11
      IF (IDWA. NE.400) GO TO 11
      IF (IDWB.NE.4RCLER) GO TO 11


8.    CALL GITCON (NCON, 300, 300)
      CALL GITIMV (NCON, IFENDER)
```

```
 9.     CALL GITCON (NCON, 100, 200)
        CALL GITMMV (NCON, MAD (3))


10.     CALL GIMACE (MAD(1), MAD(2), MAD(3))
        CALL GIERAS (LWHEEL)


11.     CALL GICNRL (NCON)


12.     CALL GURSET (-600, 0, 103B, IBUF, NBYTE, MBYTE)
        CALL GUARCG (1, -700, 0, -600, 0, -600, 0,-0,IBUF,
       1     NBYTE, MBYTE)
        CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 1)
        CALL GICOPY (IDDAD, NCON, 500, -200, 103B, IDAD1,1)
        CALL GIABRT (NCON, 14HMOVE COMPLETED, 14)
        END
```



```
13.         CALL GILPKY (NCON, 8)
            CALL GIBUT (0, NCON, IDDAD, IDDT)
            CALL GITCOF (NCON, IHNEW, IVNEW)
            IF (IABS (IHNEW -IH).GT.100) GO TO 20
            IF (IABS (IVNEW-IV).GT.100) GO TO 20
                 .
                 .
                 .
         20 IH=IHNEW           $      IV=IVNEW
                 .
                 .
                 .
```

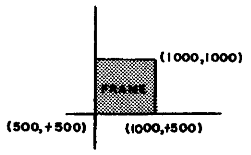
SECTION 4

```
 1.     a.      DO 10 I = 1, 4
                N = 2 ** (I-1)
            10  CALL GIMASK (NCON, 0, N, N)
                CALL GIMASK (NCON, 0, 16, 14)
                      .
                      .
                      .
                CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 2)

        b.      as b) in problem 11 with
                CALL GIMASK (NCON, 0, 16, 14)
                      .
                      .
                      .
                CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 2)


 2.     CALL GICNJB (NCON)
        CALL GICLR (NCON)
        CALL GICNRL (NCON)
```

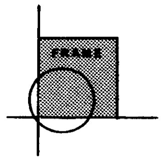
```
3.    CALL GIFID (NCON, IDDT, IDDC, IDWA, IDWB, IH, IV)


4.    CALL GIFSID (NCON, 1, IDDT, IDDC, IDWA, IDWB, IH, IV)


5.    DIMENSION IDDT (16), IDDC (16), IDWA (16), IDWB (16)
      N=16
      CALL GIBUT (0, NCON, IDDT)
      CALL GIFID (NCON, IDDTP, IDDCP, IDWAP, IDWBP)
      IF (IDDTP.NE.IDDTP) CALL USEP
      CALL GIFSID (NCON, N, IDDT, IDDC, IDWA, IDWB)
      IF (N. NE.0.AND.IDDT(1).NE.0) CALL USES
```

SECTION 5

```
1.    CALL GFONTA (NCON, 600, 600, ID1, ID2, ID3)


2.    CALL GIERAS (ID1, ID2, ID3)


3.    CALL GULINE (750, 750, 1000, 1000, AX, AY, BX, BY,
     1     KSHOW, IH1, IV1, IH2, IV2)
```



```
4.    H1=IH(1) $H2=IH(2) $V1=IV(1) $V2=IV(2)
      CALL GULINE (0, 0, 1434, 1434, H1, V1, H2, V2, KSHOW,
     1     IH1, IV(1), IH(2), IV(2)
      IF (KSHOW.EQ.0) GO TO 100
      CALL GURSET (IH(1), IV(1), 3, IBUF, NBYTE, MBYTE)
      CALL GUSEGS (IH(1), IV(1), IH(2), IV(2), IBEAM,
     1     ISTYLE, IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 1)
           .
           .
           .
  100 CONTINUE


5.    DIMENSION IH1(5), IH2(5), IV1(5), IV2(5)
      HC=IHCNTR  $VC=IVCNTR $H1=H2=IRAD+IHCNTR $V1=V2=IVCNTR
      CALL GUARC (0, 0, 1434, 1434, HC, VC, H1, V1, H2, V2,
     1     KSHOW, IHC, IVC, IH1, IV1, IH2, IV2)
      IF (KSHOW.EQ.0) GO TO 100
      CALL GURSET (IH1(1), IV1(1), 3, IBUF, NBYTE, MBYTE)
      CALL GUARCG (KSHOW, IHC, IVC, IH1, IV1, IH2, IV2,
     1     ISTYLE, IBUF, NBYTE, MBYTE)
      CALL GIDISP (NCON, IBUF, NBYTE, IDDAD, 1)


6.    CALL GUARC (750, 750, 1000, 1000, 0, 0, H1, V1, H2,
     1     V2, KSHOW, IHC, IVC, IH1, IV1, IH2, IV2)
```



7.
```
           CALL GIEOM (NCON, 1R;, 8, 66)
           N=10
     1000  CALL GIANS (NCON, N, -1000, 700)
     1100  CALL GIBUT (0, NCON, IDDT, IDDT)
           IF(IDDC.NE.66) GO TO 1100
           CALL GIANE (NCON, N, IBCD)
        C  IBCD(1) CONTAINS 1-9 LEFT JUSTIFIED HOLLERITH
          1     CHARACTERS
           IF(N.EQ.1) GO TO 1000
```

8.
```
     CALL GURSET (-35, 0, 103B, IBUF, NBYTE, MBYTE)
     CALL GUAN (IBCD, N, IBUF, NBYTE, MBYTE)
     CALL GIDISP (NCON, IBUF, NBYTE, IDAD, 2)
```

9.
```
     CALL GIKYBD(1, 2, 125)
```

10.
```
     CALL AETSKC(4LCUBE)
```

or

```
     CALL AETSKC(4HCUBE)
```

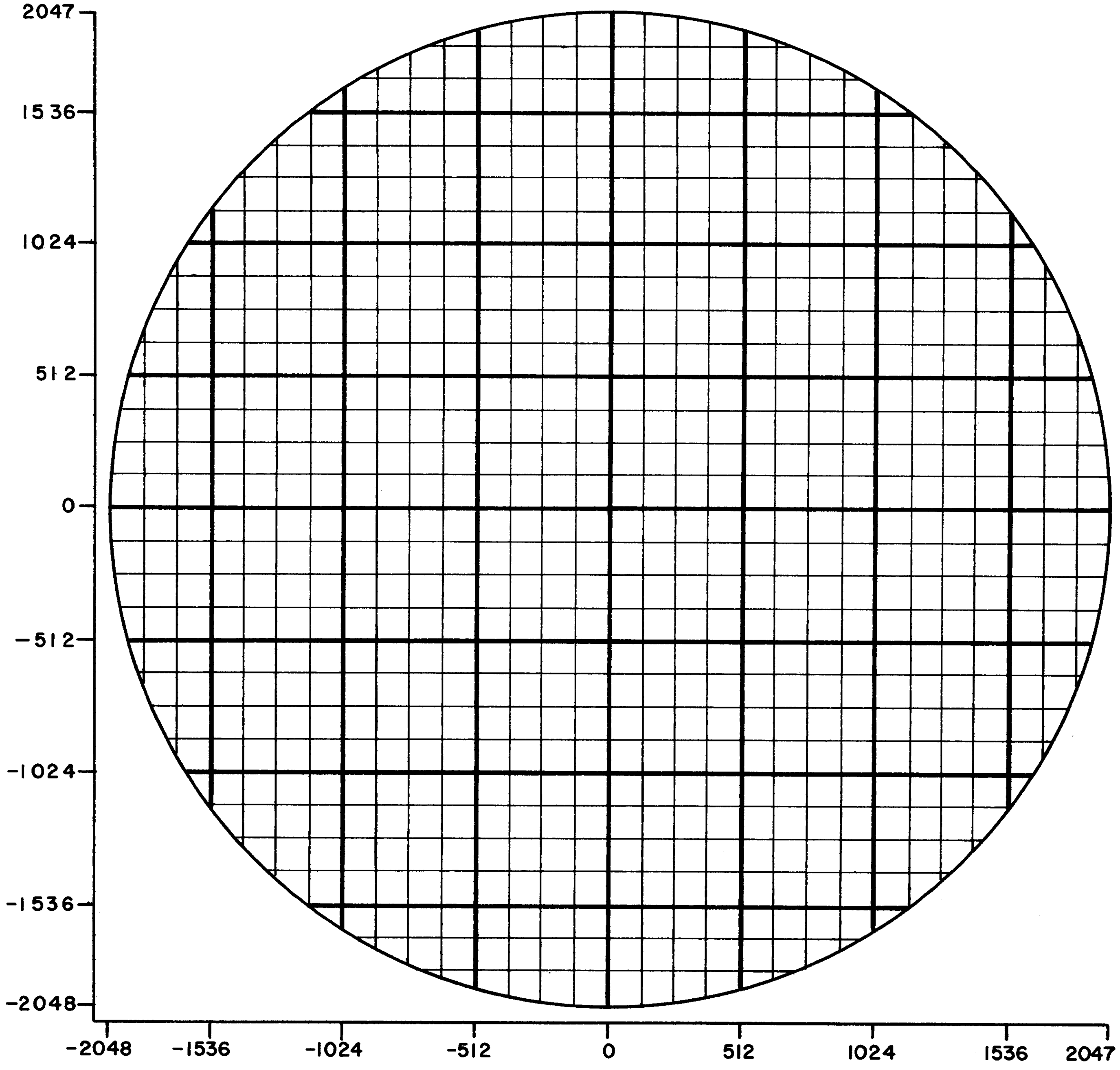
DISPLAY GRID WORKSHEET G

The following pages contain blank 274 display grid coordinate figures, for use in laying out screen displays.

Program Name

____

IGS DISPLAY GRID
WORK SHEET



INDEX

COMMENT SHEET

MANUAL TITLE 6000 Series 274 Interactive Graphics System User's Guide

PUBLICATION NO. 44629300 **REVISION** D

FROM: NAME: ____________

BUSINESS
ADDRESS: ____________

COMMENTS:

This form is not intended for use as an order blank. Your evaluation of this manual is welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be noted below. Please include page number references and fill in the publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer Engineers are urged to use the TAR.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Systems Publications

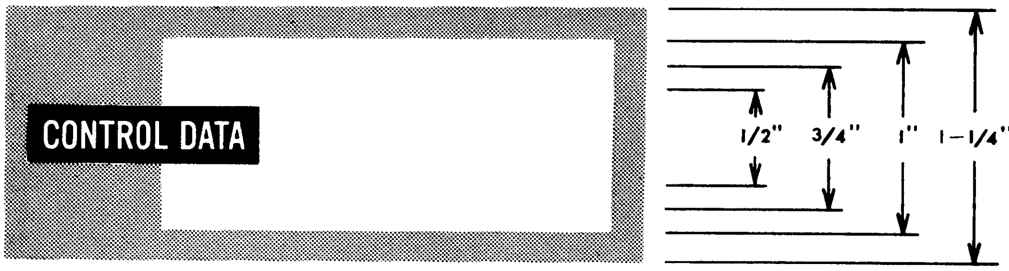
215 Moffett Park Drive

Sunnyvale, California 94086

CUT ON THIS LINE

FOLD FOLD

STAPLE STAPLE



→→ CUT OUT FOR USE AS LOOSE-LEAF BINDER TITLE TAB

274 INTERACTIVE GRAPHICS SYSTEM USERS GUIDE

Pub. No. 44629300

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN, 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

PRINTED IN U.S.A.