SLAC USER SERVICES

MASTER COPY
DO NOT REMOVE

CGTM NO. 80
ROBERT C. BEACH
DECEMBER 1969

REVISED:
MARCH 1971

THE SLAC SCOPE PACKAGE FOR THE IDIOM

A COLLECTION OF PL/1 PROCEDURES WHICH MAY BE USED TO CONTROL THE IDIOM DISPLAY CONSOLE

COMPUTATION GROUP
STANFORD LINEAR ACCELERATOR CENTER

TABLE OF CONTENTS

	SECTION
AN INTRODUCTION TO THE IDIOM SCOPE PACKAGE.	
A BRIEF DESCRIPTION OF THE HARDWARE.	1.1
A BRIEF DESCRIPTION OF THE PROCEDURES.	1.2
A BRIEF DESCRIPTION OF THE 620/I PROGRAM.	1.3
SYNCHRONIZING THE DISPLAY WITH THE SLAC MOVIE CAMERA.	1.4
SYNCHRONIZING THE DISPLAY WITH THE SLAC 3-D VIEWER.	1.5
SUGGESTIONS FOR EFFICIENT USE OF THE PACKAGE.	1.6
DETAILED DESCRIPTION OF THE PROCEDURES.	
INITIALIZATION AND TERMINATION PROCEDURES.	2.1
PROCEDURE IDOPEN	2.1.1
PROCEDURE IDCLOSE	2.1.2
ELEMENT GENERATION PROCEDURES.	2.2
PROCEDURE IDPOS	2.2.1
PROCEDURE IDTEXT	2.2.2
PROCEDURE IDPTLN	2.2.3
PROCEDURE IDEXELT	2.2.4
PROCEDURE IDDBUF	2.2.5
PROCEDURE IDSCR	2.2.6
PROCEDURE IDORDER	2.2.7
ELEMENT AND SET MANIPULATION PROCEDURES.	2.3
PROCEDURE IDEPUT	2.3.1
PROCEDURE IDEGET	2.3.2
PROCEDURE IDSET	2.3.3
PROCEDURE IDMODX	2.3.4
PROCEDURE IDLINK	2.3.5
PROCEDURE IDAPPLY	2.3.6
PROCEDURE IDMEMB	2.3.7
PROCEDURE IDDATA	2.3.8
PROCEDURE IDAUXL	2.3.9
ELEMENT DATA RECOVERY PROCEDURES.	2.4
PROCEDURE IDXTTYB	2.4.1
PROCEDURE IDXLPDB	2.4.2
INTERRUPT PROCESSING PROCEDURES.	2.5
PROCEDURE IDEATTN	2.5.1
PROCEDURE IDSATTN	2.5.2
PROCEDURE IDDATTN	2.5.3
PROCEDURE IDRATTN	2.5.4
TRACKING CROSS CONTROL PROCEDURES.	2.6
PROCEDURE IDTCPUT	2.6.1
PROCEDURE IDTCRMV	2.6.2
PROCEDURE IDTCPOS	2.6.3
PROCEDURE IDTCDEL	2.6.4
PROCEDURE IDTCDMD	2.6.5

TABLE OF CONTENTS (CONTINUED)

	SECTION
MISCELLANEOUS PROCEDURES.	2.7
PROCEDURE IDBGEN	2.7.1
PROCEDURE IDBFRAM	2.7.2
PROCEDURE IDLITES	2.7.3
PROCEDURE IDTRACE	2.7.4
PROCEDURE IDESUBR	2.7.5
PROCEDURE IDSHUTR	2.7.6
PROCEDURE IDEPRNT	2.7.7
PROCEDURES IDEPCH1 AND IDEPCHN	2.7.8
PROCEDURE IDSYMB	2.7.9
SECOND LEVEL PROCEDURES.	2.8
PROCEDURES IDERRP AND IDERRI	2.8.1
PROCEDURES IDREAD, IDWRITE, AND IDWRIT2	2.8.2
PROCEDURE IDEOPR	2.8.3
PROCEDURE IDATTN	2.8.4
PROCEDURE IDCBENT	2.8.5
PROCEDURES IDCBGET AND IDCBPUT	2.8.6
PROCEDURES LKOPEN, LKREAD, LKWRITE, LKCLOSE, AND LKDATTN	2.8.7
PROCEDURE ECBCTRL	2.8.8
PROCEDURE TIMCTRL	2.8.9
PROCEDURES AXSAVE AND AXCALL	2.8.10
PROCEDURES EBTOID AND IDTOEB	2.8.11
PROCEDURE VCONC	2.8.12
PROGRAMMING EXAMPLES.	
AN AREA AND ARC LENGTH CALCULATOR.	3.1
INFORMATION ABOUT THE INTERNAL OPERATION OF THE PACKAGE.	
PROCEDURE LENGTHS AND CROSS REFERENCE.	4.1
THE FORMAT AND USE OF THE CONTROL BLOCKS.	4.2

AN INTRODUCTION TO THE IDIIOM SCOPE PACKAGE.

The group of procedures described herein is designed to allow the PL/1 programmer to write programs using the IDIIOM display console at SLAC with a minimum of effort. The SYSTEM/360-620/I-IDIIOM combination is an extremely versatile combination and it is unlikely that any high level programming system could take advantage of all of the options available. It is hoped however that this package will allow a large class of interactive display problems to be attacked without having the programmer become lost in a jungle of details.

A BRIEF DESCRIPTION OF THE HARDWARE. SECTION 1.1

The IDIIOM consists of a Varian 620/I computer with 8192 words of 16 bits each, a teletype, and a display console made by Information Displays Incorporated. The display console includes a CRT with a usable area of 13 inches by 13 inches, a light pen, and a set of 32 function keys. When the IDIIOM is operating, the 620/I memory will contain a program for the 620/I to execute, and a program (display file) for the CRT to execute. The instructions (orders) in the display file may include orders to display characters or straight line segments, perform unconditional or subroutine jumps, or interrupt the 620/I. The 620/I instruction set includes, in addition to the usual set for a modern small computer, instructions to start and stop the CRT in its execution of the display file and instructions to read and reset registers associated with the display operation.

The IDIIOM and the SYSTEM/360 are connected together through an IBM 2701 Data Adapter Unit. Information may be transmitted either way through this link. In addition the 620/I has the ability to send an attention interrupt to the SYSTEM/360. The SYSTEM/360 is not able to interrupt the IDIIOM however.

The CRT screen on the IDIIOM display console has an imaginary grid of 1024 by 1024 raster units imposed on it. A point may be plotted at any addressable location and a line may be drawn between any two addressable locations. The lower left corner of the display has coordinates (0,0) and the upper right has coordinates (1023,1023). Characters may be plotted in four sizes, small, medium, large, and very large. These characters may be plotted either horizontally, or vertically (rotated 90 degrees counter-clockwise). The coordinates given for plotting a character specify the lower left corner of the character. The following table gives more information on the plotting of characters. In this table, the between-character dimensions are set by the hardware while the between-lines dimensions are set by the programmer and the given valves represent suggestions.

CHARACTERISTIC	SMALL	MEDIUM	LARGE	V. LARGE
CHARACTERS PER LINE	128	73	48	36
LINES PER DISPLAY	85	48	34	24
CHARACTER SPACING (RASTER UNITS)	8	14	21	28
LINE SPACING (RASTER UNITS)	12	21	30	42

Information may be displayed in any of four intensity levels. Lines may have any of four line structures (solid, dashed, dot-dashed, or dots). Displayed information may be in a blinking mode or a steady mode. In addition the light pen can be made sensitive or insensitive to displayed information.

A BRIEF DESCRIPTION OF THE PROCEDURES. SECTION 1.2

The user of this package will not have to do any coding for the 620/I or the display scope. All of the programming is done for the SYSTEM/360 in the PL/1 programming language. To the user it will appear that the SYSTEM/360 is in control, and the IDIIOM is an on-line display device.

The basic items which can be displayed when using this package are called display elements. Procedures are supplied to construct elements which will display information on the CRT. Other procedures are supplied to transmit these elements to and from the IDIICM. In addition, related elements may be grouped together into sets. A set may consist of elements and other sets.

An element that has been transmitted to the IDIIOM may be displayed as transmitted, or it may be executed by another element. In this second case the element-subroutine will usually not contain any absolute positioning orders. Instead the calling element will contain absolute positioning orders followed by a call to the element-subroutine. In this way many copies of an element may appear on the screen at one time.

In addition to the type of element which causes a static display to appear on the CRT, there are some special element types available. These special elements are:

1. A teletype input buffer: Typing on the teletype or the keyboard will change the characters in the element. This can be used to enter textual information into the system from the console.
2. An element may be positioned relative to the tracking cross: The console operator may therefore move that part of the display around on the screen with the light pen.
3. A light pen drawing buffer: The light pen may be used to draw freehand figures on the CRT. The orders for the figure are put into the light pen drawing buffer. These orders may be read and processed by the SYSTEM/360.

A BRIEF DESCRIPTION OF THE PROCEDURES (CONTINUED). SECTION 1.2

Procedures are provided which give the user of this package the ability to specify how the IDIIOM is to process interrupts from the function keyboard, light pen, or other devices. These procedures may be used to specify that an interrupt is to be sent to the SYSTEM/360 if such an action occurs, or that the action is to be ignored. In the first case the interrupt is said to be enabled, and in the second case it is said to be disabled.

Errors detected by the procedures are reported to the calling program in two distinct ways. In some cases a parameter in the calling sequence is set to identify the problem. These errors tend to be the type for which the programmer has a chance of easily recovering. The second type of error reporting mechanism is used on errors which probably represent coding errors. In this case an error processing procedure is called which raises the IDERROR condition. If the programmer has coded:

```
CN CONDITION(IDERROR) BEGIN; ... END;
```

He will obtain control from the error processing procedure, otherwise an error message is printed and the program is terminated with a core dump.

A BRIEF DESCRIPTION OF THE 620/I PROGRAM. SECTION 1.3

The 620/I part of this scope package must have been loaded into the 620/I before a SYSTEM/360 program using these procedures begins executing. The procedures in this package may then communicate with the 620/I program. They send the 620/I elements for display, information on how interrupts are to be processed, and other control information. The 620/I program maintains the display file, and monitors all of the devices (teletype, light pen, function buttons, etc.) attached to the 620/I-IDIIOM system. The 620/I will send interrupts to the SYSTEM/360 when necessary.

The normal procedure to be followed to execute a program using the procedures in this package is as follows:

1. Load the 620/I program into the 620/I.
2. Set the sense switches on the 620/I to indicate the mode of operation for the 620/I. The proper position for the normal interactive display mode is (Down-Down-Down).
3. Begin execution in the 620/I at location 1000 (OCTAL).
4. Begin execution of the SYSTEM/360 program.

The sense switches on the 620/I are used to control the mode of operation of the 620/I. The normal interactive display mode is indicated by having all of the sense switches in the down position. The meaning of other settings of the sense switches is covered in the following sections. To change the setting of the sense switches while a SYSTEM/360 program is running, the following procedure should be used:

1. Bring the program to a state where no I/O between SYSTEM/360 and the 620/I is in progress. The SYSTEM/360 should be in the wait state.
2. Halt the 620/I.
3. Change the sense switch setting.
4. Re-start the 620/I at location 1002 (OCTAL).

If the IDIIOM has a display on the screen at the time that the sense switch setting is being changed, then steps 2 through 4 may be replaced by:

2'. Change the sense switch setting.
3'. Press the HALT button on the IDIIOM console twice. The first push on the HALT button will stop the display. The second push will cause the sense switches to be read, and the display to be re-started.

When the 620/I program is first started at location 1000 (OCTAL), a test pattern will appear on the screen. When the SYSTEM/360 program begins executing (and calls procedure IDOPEN) the test pattern will disappear from the screen and the 620/I will be waiting for the SYSTEM/360 to give it elements to display. When the SYSTEM/360 program is finished (and calls procedure IDCLOSE) a set of statistics for the run is displayed on the screen. These statistics include high water marks for usage of 620/I core and counts of all I/O activity between SYSTEM/360 and 620/I. The console operator may restore the test pattern to the screen by pressing the HALT button on the IDIIOM console twice.

SYCHRONIZING THE DISPLAY WITH THE SLAC MOVIE CAMERA. SECTION 1.4

The SLAC movie camera consists of a 16 millimeter, model 16M ARRIFLEX motion picture camera with an animation motor. An interface between the movie camera and the 620/I has been designed and built at SLAC. Through this interface, the 620/I can control the animation motor and sense when the shutter is open.

The movie camera may be operated in any of three movie-making modes with this package. The mode is selected by the position of the three sense switches on the 620/I console. These three modes and the corresponding position of the sense switches are:

1. Frame Synchronized Mode (Down-Down-Up): In this mode the movie camera runs at its normal speed of 24 frames per second and the 620/I continually senses the shutter open signal from the camera. When the shutter initially opens, the 620/I will start the display. After going through the display file once, the display is turned off until the next time the shutter opens. This mode may therefore be used to take flicker-free movies of any existing interactive display program.

2. Single Pulse Animation Mode (Down-Up-Down): In this mode the 620/I sends signals to the camera telling it when to open its shutter. After the shutter has opened, the display will be started. The length of time that the shutter remains open is controlled by a switch on the animation motor. This switch may be set to give exposure times of 1/8, 1/4, and 1/2 seconds. A single pulse animation program is written in the following manner:
   a. The 'FRAM' interrupt should be enabled and the regeneration count should be set (usually to 1) with procedure IDNFRAM.
   b. The elements for a single frame should be transmitted to the IDIIOM and the display should be started by calling IDRGEN with the 'STRT' option.
   c. The occurrence of the 'FRAM' interrupt indicates that one frame of movie film has been exposed.

   This type of animation program may be run and checked out as a normal interactive display program. The only difference will be that the motion on the screen will have a different speed than that of the final movie. In the interactive mode the picture may be changing at up to 60 times a second (it will be less than this because the transmission of information between SYSTEM/360 and 620/I uses some of the available time), while the final movie runs at 24 frames per second.
3. Double Pulse Animation Mode (Down-Up-Up); This mode is a generalization of the previous mode. It covers the case where a single frame may consist of many display files. Thus, a single frame of movie film may contain a more complex picture than can be viewed in the normal interactive mode. A double pulse animation program is written in the following manner:
   a. The 'FRAM' interrupt should be enabled and the regeneration count should be set.
   b. The shutter should be opened with procedure IDSHUTR.
   c. The elements for the display files should be transmitted and the display should be started by calling IDRGEN with the 'STRT' option.
   d. The occurrence of the 'FRAM' interrupt indicates that the display file has been exposed and the previous step may be repeated.
   e. When all display files have been exposed, the shutter should be closed with procedure IDSHUTR.

   This type of animation may also be run and checked out in the normal interactive mode. Calls to IDSHUTR will be ignored in the 620/I if it is not in the double pulse animation mode. However, the pictures on the CRT may not be very meaningful when a double pulse animation program is run in the normal interactive mode.

SYNCHRONIZING THE DISPLAY WITH THE SLAC 3-D VIEWER — SECTION 1.5

The SLAC 3-D viewer is a device which contains a motor driven rotating disk. The disk contains clear areas and opaque areas. When the user looks through this device, the left and right eyes will alternately be blocked by the opaque areas on the disk.

An interface has been built at SLAC which allows the 620/I to sense when the left or right eye has a clear view. By flashing different images on the screen when each eye is open, it is possible to present the user with a stereoscopic picture.

A program which uses the 3-D viewer should obey the following conventions:

1. Elements which are to be presented to the left eye only should have element identifications which are less than -1000.
2. Elements which are to be presented to the right eye only should have element identifications which are greater than +1000.
3. Elements which are to be presented to both eyes should have element identifications between -1000 and +1000 (inclusive).

In addition the sense switches should be put into the (Up-Down-Down) position. This setting of the sense switches signals the 620/I program that it is to synchronize the display with the 3-D viewer.

SUGGESTIONS FOR EFFICIENT USE OF THE PACKAGE. — SECTION 1.6

There are many procedures in this package and the possibilities of using something inefficiently or incorrectly are great. Following are some suggestions which may help the user avoid some of these pitfalls:

1. Try to keep the number of elements to a minimum. It is more efficient to have a small number of large elements than a large number of small elements. When using the 3-D viewer, it is especially important to keep the number of elements to a minimum.
2. The values of all of the parameters in IDOPEN should be kept to a minimum. Unnecessarily large values will cause a large block of core to be wasted.
3. There are times when it is useful to enable an interrupt with the 'KRUN' disposition (for example the 'TCHN' interrupt). However it is better in most cases to enable interrupts with the 'STOP' disposition and then restart the display (with procedure IDRGEN) after each interrupt. This will assure that the SYSTEM/360 and the 620/I do not get out of phase.

4. Animation may be performed in an interactive display program by transmitting a picture to the 620/I, waiting a specified time in the SYSTEM/360 (using the PL/1 DELAY statement or a positive value in the TIME parameter of procedure IDRATTN) or counting regeneration cycles in the 620/I (using procedure IDNFRAM and the 'FRAM' interrupt), and then transmitting another picture. Two things may be done to minimize the flicker when operating in this manner. First, the variable part of the picture should be consolidated into a single element. This will minimize the number of records transmitted between the 620/I and the SYSTEM/360. Second, element-subroutines should not be used at this time. If the package has been notified that element-subroutines are not to be processed (by calling procedure IDESUBR) the work load in the 620/I is reduced.

This section describes the initialization and termination procedures to be used in conjunction with the other procedures in this package. These procedures and their functions are:

PROCEDURE	FUNCTION
IDOPEN	Initialize the IDIIOM and put it under the control of the other procedures in this package.
IDCLOSE	Terminate the use of the IDIIOM by the procedures in this package.

```
*************************************************************************
*                                                                       *
*  This procedure opens the DCB for the link.  It also allocates        *
*  some in-core blocks which are used to accomplish link I/O and        *
*  process interrupts.  In addition it sends information to the         *
*  link which causes the IDIIOM to initialize itself.  At this          *
*  time the test pattern will disappear from the CRT screen.  This      *
*  procedure must be called before any other procedures in the          *
*  IDIIOM scope package are called.                                     *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDOPEN(NENT,NSET,NAUXL);                                      *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    NENT     The maximum number of sets and elements which will       *
*             exist at any one time.                                    *
*    NSET     The maximum number of sets that a set or element may     *
*             belong to.                                                *
*    NAUXL    The number of characters in a string of information      *
*             which is associated with each set or element.            *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDOPEN ENTRY(FIXED BINARY,FIXED BINARY,FIXED BINARY);      *
*                                                                       *
*  The error codes produced by this procedure are:                      *
*    1 - At least one of the input parameters contains an invalid       *
*        value.                                                         *
*                                                                       *
*************************************************************************
```



```
*************************************************************************
*                                                                       *
*  This procedure closes the DCB for the link.  It also frees           *
*  some in-core blocks which are used to accomplish link I/O and        *
*  process interrupts.  In addition it sends information to the         *
*  IDIIOM which causes the summary statistics for the run to be         *
*  displayed on the CRT screen.  Once this procedure is called,         *
*  no use may be made of the other procedures in the IDIIOM scope       *
*  package until the link is re-opened.                                 *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDCLOSE;                                                      *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDCLOSE ENTRY;                                             *
*                                                                       *
*************************************************************************
```

This section describes procedures which may be used to generate orders for the IDIIOM. The orders that are generated are placed in a 'display element'. A display element is a varying length character string. To generate a complete display element, the length should first be set to zero, then these procedures may be called to add display orders to the element. Finally the display element may be transmitted to the IDIIOM where it may be displayed. The procedures described in this section, and their functions are:

PROCEDURE	FUNCTION
IDPOS	Add beam positioning orders to a display element.
IDTEXT	Add text generation orders to a display element.
IDPTLN	Add point or line generation orders to a display element.
IDEXELT	Add orders to execute another element to a display element.
IDDBUF	Add orders to form a light pen drawing buffer (LPDB) element.
IDSCR	Add an SCR order to a display element. The programmer will not normally have to use this procedure.
IDORDER	Add any IDIIOM display order to a display element. The programmer will not normally have to use this procedure.

The following example will illustrate the use of these procedures. The display element which is created consists of the characters 'HI THERE' contained is a rectangle. The element is positioned near the center of the screen.

```
DECLARE XRECT(5) FIXED BINARY STATIC INITIAL(417,605,605,417,417);
DECLARE YRECT(5) FIXED BINARY STATIC INITIAL(484,484,536,536,484);
DECLARE ELEMENT CHARACTER(500) VARYING;
    ELEMENT='';       /*  SET ELEMENT LENGTH TO ZERO.  */
    CALL IDPOS('ABSL',XRECT(1)+10,YRECT(1)+10,ELEMENT,I);
    CALL IDTEXT('LARGHORZ','HI THERE',ELEMENT,I);
    CALL IDPOS('ABSL',XRECT(1),YRECT(1),ELEMENT,I);
    CALL IDPTLN('LINECOMP',XRECT,YRECT,101B,'1'B,ELEMENT,I);
```

If the LPCR mode of the light pen is to be used, then the elements must obey the following conventions:

1. The element must not contain any SCR instructions which change the light pen detectability. Other SCR's may be present, but the programmer will find it difficult to make any changes to these instructions.
2. Either (1) the first orders are absolute positioning orders or (2) the element is constructed in such a manner that it leaves the beam positioned in the same place it was at the beginning of the element.

```
*************************************************************************
*                                                                       *
*  This procedure generates IDIIOM display orders to re-position        *
*  the CRT beam.  The orders are inserted into a varying character      *
*  string.                                                              *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDPOS(STRING,X,Y,ELEMENT,IEXIT);                              *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    STRING   Any of ('ABSL', 'RELT', 'XRYA', 'XAYR') to indicate       *
*             absolute or relative positioning of the beam in the       *
*             X and Y directions.  If this parameter cannot be          *
*             recognized, 'ABSL' is assumed.                            *
*    X        X positioning coordinate.                                 *
*    Y        Y positioning coordinate.                                 *
*    ELEMENT  A varying character string which will have the            *
*             display orders added to it.                               *
*    IEXIT    A zero if the generated orders all fit into ELEMENT,      *
*             or the number of bytes that did not fit.                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDPOS ENTRY(CHARACTER(*),FIXED BINARY,FIXED BINARY,        *
*      CHARACTER(*) VARYING,FIXED BINARY);                              *
*                                                                       *
*************************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure generates IDIIOM display orders to display text       *
*  data.  The orders are inserted into a varying character string.      *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDTEXT(STRING1,STRING2,ELEMENT,IEXIT);                        *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    STRING1  Any of ('SMAL', 'MEDM', 'LARG', 'VLRG'), ('HORZ',         *
*             'VERT') concatenated together.  If this parameter         *
*             cannot be recognized, 'MEDM' and 'HORZ' is assumed.       *
*    STRING2  A character string containing the characters to be        *
*             added to element.  These characters should be in          *
*             SYSTEM/360 code.                                          *
*    ELEMENT  A varying character string which will have the            *
*             display orders added to it.                               *
*    IEXIT    A zero if the generated orders all fit into ELEMENT,      *
*             or the number of bytes that did not fit.                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDTEXT ENTRY(CHARACTER(*),CHARACTER(*),                    *
*      CHARACTER(*) VARYING,FIXED BINARY);                              *
*                                                                       *
*************************************************************************
```

```
*****************************************************************************
*                                                                           *
*  This procedure generates IDIIOM display orders to display                *
*  points and lines.  The orders are inserted into a varying                *
*  character string.                                                        *
*                                                                           *
*  Tne calling sequence is:                                                 *
*    CALL IDPTLN(STRING,XARRAY,YARRAY,NARRAY,BLKBITS,ELEMENT,               *
*      IEXIT);                                                              *
*                                                                           *
*  The parameters in the calling sequence are:                              *
*    STRING   Any of ('PNTS', 'LINE'), ('COMP', 'NCMP') concaten-           *
*             ated together.  If this parameter cannot be recog-            *
*             nized, 'PNTS' and 'COMP' are assumed.  'COMP' means           *
*             that the orders should be compact, that is, use a             *
*             minimum of space.  In this case short point, vector,          *
*             and blank orders will be used whenever possible.              *
*             'NCMP' means that all beam motion will be done with           *
*             a long deferred order followed by a long point,               *
*             vector, or blank order.                                       *
*    XARRAY   The array of X coordinates.                                   *
*    YARRAy   The array of Y coordinates.                                   *
*    NARRAY   The number of points in the coordinate arrays.  The           *
*             first point is (XARRAY(1),YARRAY(1)) and the last             *
*             point is (XARRAY(NARRAY),YARRAY(NARRAY)).  The                *
*             orders that are generated are orders to move from             *
*             the first point to the second, from the second to             *
*             the third, etc.  No absolute positioning orders are           *
*             generated.  Also note that this procedure results in          *
*             the generation of orders for NARRAY points or                 *
*             (NARRAY-1) lines.                                             *
*    BLKBITS  The blanking bits.  The bits in this bit string are           *
*             used cyclically.  A '1'B causes the line or point to          *
*             be intensified while a '0'B causes it to be blanked.          *
*    ELEMENT  A varying character string which will have the                *
*             display orders added to it.                                   *
*    IEXIT    A zero if the generated orders all fit into ELEMENT,          *
*             or the number of bytes that did not fit.                      *
*                                                                           *
*  The entry declaration for this procedure is:                             *
*    DECLARE IDPTLN ENTRY(CHARACTER(*),(*) FIXED BINARY, (*)                *
*      FIXED BINARY,FIXED BINARY,BIT(*),CHARACTER(*) VARYING,               *
*      FIXED BINARY);                                                       *
*                                                                           *
*****************************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure generates orders to execute another element.          *
*  The orders are inserted into a varying character string.  Thus       *
*  an element may be treated as a subroutine and other elements         *
*  may execute such an element.  Calls to these element-                *
*  subroutines may be nested to any depth.  The actual orders           *
*  generated consist of special flags which the IDIIOM will             *
*  recognize when the element is transmitted to the IDIIOM.  The        *
*  IDIIOM then resolves these flags into actual calls to element-       *
*  subroutines.  Note that this checking for element-subroutines        *
*  is usually turned off.  See procedure IDESUBR for details.           *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDEXELT(IELT,ELEMENT,IEXIT);                                  *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    IELT     The identification of the element to be executed.         *
*    ELEMENT  A varying character string which will have the            *
*             display orders added to it.                               *
*    IEXIT    A zero if the generated orders all fit into ELEMENT,      *
*             or the number of bytes that did not fit.                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDEXELT ENTRY(FIXED BINARY,CHARACTER(*) VARYING,           *
*      FIXED BINARY);                                                   *
*                                                                       *
*************************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure generates the orders necessary to create a light  *
*  pen drawing buffer (LPDB) element.  The orders are inserted      *
*  into a varying character string.                             *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDDBUF(NBYTES,ELEMENT,IEXIT);                         *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    NBYTES   The length of the drawing buffer in bytes.  The   *
*             start of each curve in the buffer requires 6 bytes.  *
*             Each straight line segment less than 32 raster units  *
*             long requires 2 bytes and longer segments require 4   *
*             bytes.  If NBYTES is odd, it is rounded up to the     *
*             nearest even number, if it is less than 10 it is      *
*             set to 10.                                        *
*    ELEMENT  A varying character string which will have the    *
*             display orders added to it.                       *
*    IEXIT    A zero if the generated orders all fit into ELEMENT,  *
*             or the number of bytes that did not fit.          *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDDBUF ENTRY(FIXED BINARY,CHARACTER(*) VARYING,    *
*      FIXED BINARY);                                           *
*                                                               *
*****************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure generates an SCR display order for the IDIIOM.        *
*  The order is inserted into a varying character string.               *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDSCR(STRING,ELEMENT,IEXIT);                                  *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    STRING   Any of ('SOLD', 'DASH', 'DDSH', 'DOTS'), ('DIMM',         *
*             'MEDM', 'BRIT', 'VBRT'), ('WINK', 'STDY'), ('DETC',       *
*             'NDET') concatenated together.  If more than one          *
*             item is chosen from a group, the first item is used.      *
*    ELEMENT  A varying character string which will have the            *
*             display order added to it.                                *
*    IEXIT    A zero if the generated order fits into ELEMENT,          *
*             or the number of bytes that did not fit.                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDSCR ENTRY(CHARACTER(*),CHARACTER(*) VARYING,             *
*      FIXED BINARY);                                                   *
*                                                                       *
*************************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to generate any of the IDIIOM     *
*  display file orders and insert them into a varying character *
*  string.  The programmer should study the internal operation of *
*  the 620/I program before using this procedure.  The use of   *
*  the index register in particular can cause the display to enter *
*  an infinite loop, and possibly damage the phosphor on the CRT *
*  screen.                                                      *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDCRDER(STRING,ARG1,ARG2,ELEMENT,IEXIT);              *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    STRING   A code for the order to be generated.  The parameters *
*             ARG1 and ARG2 provide additional information to   *
*             generate the order.  The valid values of STRING and *
*             the corresponding meaning of ARG1 and ARG2 are:   *
*               'PNXA'  Position X register (absolute).  ARG1   *
*                       contains the X value.                   *
*               'PNXR'  Position X register (relative).  ARG1   *
*                       contains the delta-X value.             *
*               'PNYA'  Position Y register (absolute).  ARG1   *
*                       contains the Y value.                   *
*               'PNYR'  Position Y register (relative).  ARG1   *
*                       contains the delta-Y value.             *
*               'LDIA'  Load index register (absolute).  ARG1   *
*                       contains the index value.               *
*               'LDIR'  Load index register (relative).  ARG1   *
*                       contains the index increment.           *
*               'LPM '  Enter line and point mode.              *
*               'SPT '  Short point.  ARG1 and ARG2 give the    *
*                       delta-X and delta-Y values.             *
*               'SLN '  Short line.  ARG1 and ARG2 give the     *
*                       delta-X and delta-Y values.             *
*               'RPT '  Repeat point.  ARG1 and ARG2 give the   *
*                       delta-X and delta-Y values.             *
*               'SBL '  Short blank.  ARG1 and ARG2 give the    *
*                       delta-X and delta-Y values.             *
*               'PNTX'  Long point (X).  ARG1 gives the delta-X *
*                       value.                                  *
*               'PNTY'  Long point (Y).  ARG1 gives the delta-Y *
*                       value.                                  *
*               'LVTX'  Long vector (X).  ARG1 gives the delta-X *
*                       value.                                  *
*               'LVTY'  Long vector (Y).  ARG1 gives the delta-Y *
*                       value.                                  *
*               'LDFX'  Long deferred (X).  ARG1 gives the delta-X *
*                       value.                                  *
*               'LDFY'  Long deferred (Y).  ARG1 gives the delta-Y *
*                       value.                                  *
```

```
*                'LBLX'  Long blank (x).  ARG1 gives the delta-X      *
*                        value.                                       *
*                'LBLY'  Long blank (Y).  ARG1 gives the delta-Y      *
*                        value.                                       *
*                'CHM '  Enter character mode.  ARG1 gives the        *
*                        character size value.                        *
*                'RCM '  Enter rotated character mode.  ARG1 gives    *
*                        the character size value.                    *
*                'CHD '  Character data.  The 6 low order bits of     *
*                        ARG1 and ARG2 are used.                      *
*                'GRM '  Enter graph mode.                            *
*                'GRDV'  Graph data (vector).  ARG1 gives the         *
*                        delta-Y value.                               *
*                'GRDP'  Graph data (point).  ARG1 gives the          *
*                        delta-Y value.                               *
*                'TIXA'  Transfer index to X register (absolute).     *
*                'TIXR'  Transfer index to X register (relative).     *
*                'NOD '  No data.                                     *
*    ARG1      The first argument for STRING.  ARG1 is ignored       *
*              for some values of STRING.                            *
*    ARG2      The second argument for STRING.  ARG2 is ignored      *
*              for some values of STRING.                            *
*    ELEMENT   A varying character string which will have the        *
*              display order added to it.                            *
*    IEXIT     A zero if the generated order fit into ELEMENT,       *
*              the number of bytes that did not fit, or a minus      *
*              one if STRING contained an invalid code.              *
*                                                                    *
*  The entry declaration for this procedure is:                      *
*    DECLARE IDORDER ENTRY(CHARACTER(4),FIXED BINARY,FIXED BINARY,   *
*      CHARACTER(*) VARYING,FIXED BINARY);                           *
*                                                                    *
**********************************************************************
```

This section describes procedures which may be used to manipulate and organize elements and transmit them to and from the IDIIOM. The basic items in a display are the elements. These elements may be grouped into sets, and these sets may in turn belong to other sets. The number of sets that a set or element may belong to was specified in the NSET parameter in procedure IDOPEN. These procedures and their functions are:

PROCEDURE	FUNCTION
IDEPUT	Transmit an element to the IDIIOM and establish its initial properties.
IDEGET	Read an element from the IDIIOM.
IDSET	Create a set. Display elements and other sets may later become members of a newly created set.
IDMODX	Delete an element or set, or modify the display properties of a element.
IDLINK	Remove a set or element from one set and insert it on another set.
IDAPPLY	Apply a procedure to each set or element which may be reached from a given set.
IDMEMB	Generate a list of sets which contain a given set or element.
IDDATA	Obtain information about a given set or element.
IDAUXL	Store and retrieve auxiliary information associated with a set or element.

***** ELEMENT TYPES AND PROPERTIES *****

When an element is transmitted to the IDIIOM its initial properties are specified. These properties include the element type and its display modes. Once an element is transmitted to the IDIIOM, its type is fixed, however its display modes may be easily changed if the element does not contain any SCR instructions. A description of the element types and display modes follows:

ELEMENT TYPES:

NORM A normal element with no special properties.

TTYB A teletype input buffer. A TTYB element must consist of absolute positioning orders generated by IDPOS followed by horizontal text orders generated by IDTEXT. The console operator may change the characters in the orders generated by IDTEXT by typing on the teletype or the keyboard. It is strongly suggested that the number of characters in a TTYB element be even.

RTTC The element will be positioned relative to the tracking cross. It should not contain any absolute positioning orders.

LPDB A light pen drawing buffer. When a LPDB element is present, the operational mode may be set to inactive, draw, or erase by procedure IDTCDMD. When the draw mode is selected, depressing the light pen switch and moving the tracking cross will cause drawing to take place. Releasing the light pen switch causes the current curve to be completed. Many curves may be put into one LPDB element. The minimum size of the line segments in the curve is set by procedure IDTCDEL. When the erase mode is selected, pointing to a line segment while the light pen switch is depressed will cause that segment to disappear (the space in the LPDB element is not reusable). When operating in either mode, enabled interrupts have priority over the LPDB operation. LPDB elements must contain only orders generated by procedure IDDBUF.

At any one time the IDIIOM may contain many NORM and TTYB elements. However only one RTTC or LPDB element can be manipulated at any one time. If a second such element is transmitted, the first element will act as a NORM element from then on.

DISPLAYABILITY MODES:

INCL The element is in the display file and it will appear on the screen if the display is regenerating.

CMIT The element is in the display file but the orders are effectively skipped. The element will not appear on the screen.

LIGHT PEN DETECTABILITY MODES:

DETC The element is detectable by the light pen.

NDET The element is not detectable by the light pen.

DISPLAY STEADYNESS MODES:

WINK The element will wink on and off when it is being displayed.

STDY The element will remain steady on the screen.

INTENSITY LEVELS:

DIMM The intensity of the element is dim.

MEDM The intensity of the element is medium.

BRIT The intensity of the element is bright.

VBRT The intensity of the element is very bright.

LINE STRUCTURES:

SOLD Lines in the element will be solid.

DASH Lines in the element will be dashed.

DDSH Lines in the element will be dot-dashed.

DOTS Lines in the element will be dotted.

The element generated in the example of section 2.2 may be transmitted to the IDIIOM by the statement:

```
CALL IDEPUT(10,'NORMINCLNDETSTDYDIMMSOLD',ELEMENT);
```

Because the second argument contains many of the default values, this call is completely equivalent to:

```
CALL IDEPUT(10,'DIMM',ELEMENT);
```

At some later time in the program, it may be desirable to change this element by making it light pen detectable and making it brighter on the screen. This may be done by:

```
CALL IDMODX(10,'DETCBRIT');
```

The 'NODV' option in the second parameter of IDEPUT is available for a special use. It is designed to be applied to elements which are to act as element-subroutines. By using this option the display modes of an element-subroutine are set by the calling element. In particular this means that one copy of an element-subroutine on the screen may be light pen detectable while another copy is not detectable.

***** TELETYPE AND KEYBOARD CONTROL WITH TELETYPE INPUT BUFFERS *****

When teletype input buffers are on the screen, one of them will have a cursor associated with it. The cursor is a vertical arrow which is positioned below the element and points to one of the characters in the element. Typing a valid alphabetic-numeric-special character on the teletype or keyboard will cause that character to appear in the position marked by cursor and the cursor to move to the next character. In addition certain special operations may be done by holding the control key down and typing an alphabetic character. These special operations are:

CURSOR CONTROL OPERATIONS:

(control)+A	Advance the cursor to the next character. (On the keyboard the FS key will also have this function.)
(control)+B	Backspace the cursor. (On the keyboard the BS key will also have this function.)
(control)+N	Move the cursor to the next teletype input buffer. It is possible to have teletype buffers on the screen and not have any cursor. If this occurs, this operation will restore the cursor.
(control)+R	Reset the cursor to the start of the teletype input buffer.

ELEMENT CHANGE OPERATIONS:

(control)+C	Clear the teletype input buffer to blanks and reset the cursor to the beginning of the teletype input buffer.
(control)+D	Delete the character at the current cursor position and move the following characters one space to the left.

(control)+I Insert a blank at the current cursor position and move the character over the cursor and all following characters one space to the right.

TAB CONTROL OPERATIONS:

(control)+S Set a tab at the current cursor position.
(control)+T Tab the cursor to the next tabbed position.
(control)+U Clear the current position of a set tab.
(control)+V Clear all set tabs.

PRINT CONTROL OPERATIONS:

(control)+O Turn the teletype printer off.
(control)+P Turn the teletype printer on. After this control operation all typed characters will be printed on the teletype. In addition the line feed and return keys will respond when printing is turned on.

After a teletype input buffer has had information inserted into it by the teletype operator it may be transmitted to the SYSTEM/360. The character codes in the transmitted element will be in IDIIOM code. To use these characters for any useful computations they will first have to be translated to SYSTEM/360 character codes.

***** CREATING AND MANIPULATING SETS AND ELEMENTS *****

When an element is transmitted to the IDIIOM, a record of that action is kept in the SYSTEM/360. Elements in the IDIIOM may be grouped into sets, and sets may contain elements and other sets. There is one set, whose identification is zero, which is always present. When a new set is created, or a new element is sent to the IDIIOM, it is made a member of set zero (actually it becomes a member of set zero NSET times, where NSET is the value of the argument supplied to IDOPEN). Set zero may not be deleted and may not be made a member of another set. Set zero is referred to as the master set in some of the following sections.

The following example will illustrate some of the properties of this set-membership relationship. This example creates two elements and two sets and links them together.

```
CALL IDOPEN(20,2,0);
CALL IDEPUT(10,'NORM',ELTA);  /*  TRANSMIT ELEMENTS.  */
CALL IDEPUT(11,'NORM',ELTB);
CALL IDSET(1);  /*  CREATE SETS.  */
CALL IDSET(2);
CALL IDLINK(10,0,1);  /*  PUT ELT 10 AND SET 2 ON SET 1.  */
CALL IDLINK(2,0,1);
CALL IDLINK(10,0,2);  /*  PUT ELT 10,11 AND SET 1 ON SET 2.  */
CALL IDLINK(11,0,2);
CALL IDLINK(1,0,2);
```

As this example illustrates, the set-membership relations may be recursive. The programmer may manipulate this relationship in any way that is meaningful to him. Procedure IDAPPLY may be used to scan and process recursive structures. For example, since all sets and elements in the above example may be reached from set 2, the statement:

```
CALL IDAPPLY(2,-1,IDMODX,'DELT');
```

will delete all of the items created in that example. Actually in this example the first parameter of IDAPPLY could just as well have been 0 or 1.

```
*****************************************************************
*                                                               *
*  This procedure will write a new element to the IDIIOM or     *
*  replace an already existing element.  The initial display    *
*  properties of the element are specified at this time.        *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDEPUT(IELT,STRING,ELEMENT);                          *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    IELT      The identification of this element.  This number *
*              must be different from the identifications of all*
*              other elements and sets that are to exist at the *
*              same time this element exists.                   *
*    STRING    Any of ('NORM', 'TTYB', 'RTTC', 'LPDB'), ('INCL',*
*              'OMIT'), ('DETC', 'NDET'), ('WINK', 'STDY'),     *
*              ('DIMM', 'MEDM', 'BRIT', 'VBRT'), ('SOLD', 'DASH',*
*              'DDSH', DOTS') concatenated together.            *
*              Except for the first two options, these represent*
*              modes set at the beginning of the element.  They may*
*              be changed by SCR orders internal to the element.*
*              The default values of STRING are 'NORM', 'INCL', *
*              'NDET', 'STDY', 'MEDM', and 'SOLD'.  These default*
*              values (except 'NORM' and 'INCL') may be suppressed*
*              by including 'NODV' in STRING.  In addition STRING*
*              may contain either 'STRT' or 'STOP' to indicate that*
*              the display is to be turned on or off after the  *
*              element is transmitted to the IDIIOM.  This same *
*              action could be done by a separate call to IDRGEN.*
*              If 'STRT' or 'STOP' is not given, then no change is*
*              made to the regeneration state of the display.   *
*    ELEMENT   A varying character string containing the orders *
*              which comprise the element.                      *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDEPUT ENTRY(FIXED BINARY,CHARACTER(*),            *
*      CHARACTER(*) VARYING);                                   *
*                                                               *
*  The error codes produced by this procedure are:              *
*    1 - A set with the given identification already exists.    *
*    2 - The number of entries specified by the NENT parameter of*
*        IDOPEN have been used up.                              *
*                                                               *
*****************************************************************
```

```
*****************************************************************************
*                                                                           *
*  This procedure may be used to read an element from the IDIION            *
*  into a varying character string.                                         *
*                                                                           *
*  The calling sequence is:                                                 *
*    CALL IDEGET(IELT,ELEMENT,ICURSOR,IEXIT);                               *
*                                                                           *
*  The parameters in the calling sequence are:                              *
*    IELT     The identification of the element to be read.                 *
*    ELEMENT  A varying character string which will receive the             *
*             element.                                                      *
*    ICURSOR  If the element is a TTYB type element containing the          *
*             cursor, then this is the index of the character               *
*             containing the cursor, otherwise it is zero.                  *
*    IEXIT    A zero indicates a sucessful read.  If ELEMENT was            *
*             not large enough to contain the element, then IEXIT           *
*             gives the number of bytes that did not fit.  If an            *
*             element with the given identification could not be            *
*             found then IEXIT is set to minus one.                         *
*                                                                           *
*  The entry declaration for this procedure is:                             *
*    DECLARE IDEGET ENTRY(FIXED BINARY,CHARACTER(*) VARYING,                *
*      FIXED BINARY,FIXED BINARY);                                          *
*                                                                           *
*****************************************************************************
```

```
*****************************************************************************
*                                                                           *
*  This procedure may be used to create a set.  Display elements            *
*  and other sets may be made members of a set.  If a set with the          *
*  given identification already exists, then no action is                   *
*  performed.                                                               *
*                                                                           *
*  The calling sequence is:                                                 *
*    CALL IDSET(ISET);                                                      *
*                                                                           *
*  The parameter in the calling sequence is:                                *
*    ISET      The identification of the newly created set.  This           *
*              number must be different from the identifications of         *
*              all other elements and sets that are to exist at the         *
*              same time this set exists.                                   *
*                                                                           *
*  The entry declaration for this procedure is:                             *
*    DECLARE IDSET ENTRY(FIXED BINARY);                                     *
*                                                                           *
*  The error codes produced by this procedure are:                          *
*    1 - An element with the given identification already exists.           *
*    2 - The number of entries specified by the MENT parameter of           *
*        IDOPEN have been used up.                                          *
*                                                                           *
*****************************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure may be used to make elementary modifications to       *
*  a set or element.  The only valid operation which may be             *
*  performed on a set is to delete it.  When this happens, any          *
*  items contained in the set will have their membership                *
*  transferred to the master set.  An element may be deleted or         *
*  may have any of its display properties changed.                      *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDMODX(ISOE,STRING);                                          *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    ISOE     The identification of the set or element to be acted     *
*             upon.  If a set or element with this identification      *
*             cannot be found, then no action is taken.                 *
*    STRING   Any of ('DELT'), ('INCL', 'OMIT'), ('DETC', 'NDET'),     *
*             ('WINK', 'STDY') ('DIMM', 'MEDM', 'BRIT', 'VBRT'),        *
*             ('SOLD', 'DASH', 'DDSH', 'DOTS') concatenated             *
*             together.  If DELT is specified, any other               *
*             parameters are ignored.                                   *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDMODX ENTRY(FIXED BINARY,CHARACTER(*));                   *
*                                                                       *
*************************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to change the membership relation *
*  between a set or element and its containing set.  Specifically,*
*  a set or element is identified and this item is removed from *
*  one set and inserted on another set.                         *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDLINK(ISOE,ISET,JSET);                               *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ISOE      The identification of the set or element whose   *
*              membership relation is to be changed.  ISOE should*
*              not reference the master set.                    *
*    ISET      The item identified by ISOE is removed from the set*
*              identified by ISET.                              *
*    JSET      The item identified by ISOE is inserted on the set*
*              identified by JSET.                              *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDLINK ENTRY(FIXED BINARY,FIXED BINARY,FIXED BINARY);*
*                                                               *
*  The error codes produced by this procedure are:              *
*    1 - The item identified by ISOE could not be found.        *
*    2 - A set identified by ISET could not be found.           *
*    3 - A set identified by JSET could not be found.           *
*    4 - The item identified by ISOE is not contained on the set*
*        identified by ISET.                                    *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to apply a given procedure to all *
*  of the sets or elements that may be accessed from a given set.*
*  The set is traversed recursively and the given procedure is  *
*  called exactly once for each set or element that is encounter-*
*  ed.  If a set or element is accessible in more than one way, *
*  it is presented to the given procedure only the first time it*
*  is encountered.  The order of presentation is that the items on*
*  a set are presented in the reverse order that they were put on*
*  the set, and then the set itself is presented.               *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDAPPLY(ISOE,LEVEL,PROC,ARG);                         *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ISOE      Normally this is the identification of the set to be*
*              traversed.  If ISOE identifies an element, then that*
*              element is presented to the given procedure.  If a*
*              set or element with this identification cannot be*
*              found, then no action is taken.                  *
*    LEVEL     If level is positive it specifies the level of   *
*              recursion to be allowed.  A value of one would cause*
*              the given set to be traversed but no recursion to*
*              occur.  If unlimited recursion is to occur, then *
*              LEVEL may be set to a negative number.           *
*    PROC      The entry point of the procedure that is to be   *
*              called for each item encountered.  The procedure *
*              statement for PROC is:                           *
*                PROC: PROCEDURE(JSOE,ARG);                     *
*              Where JSOE is the identification of the item being*
*              presented, and ARG is an argument being passed from*
*              procedure IDAPPLY.  There are some restrictions on*
*              just how much PROC can do.  These restrictions are:*
*              (1) PROC should not call IDAPPLY, and (2) PROC   *
*              should not modify the set-element structure (an  *
*              exception to this is that PROC=IDMODX with ARG=  *
*              'DELT' is permitted).                            *
*    ARG       An argument which will be passed to PROC.        *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDAPPLY ENTRY(FIXED BINARY,FIXED BINARY,ENTRY,);   *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to determine what sets a given set *
*  or element belongs to.                                       *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDMEMB(ISOE,MARRAY);                                  *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ISOE     The identification of a set or element whose set  *
*             membership is to be determined.                   *
*    MARRAY   An array which will be set to contain the identif- *
*             ications of the containing sets.  MARRAY(1) through *
*             MARRAY(NSET), where NSET is the value of the      *
*             argument supplied to IDOPEN, will be computed.    *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDMEMB ENTRY(FIXED BINARY,(*) FIXED BINARY);       *
*                                                               *
*  The error codes produced by this procedure are:              *
*    1 - The item identified by ISOE could not be found.        *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to determine if a set or element  *
*  with a given identification currently exists.  If such an item *
*  exists, its nature (set or element) is made known.  If the item *
*  is an element, the programmer may request that its display   *
*  properties also be made known.                               *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDDATA(ISOE,STRING1,STRING2);                         *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ISOE      The identification of the set or element for which *
*              the data is requested.                           *
*    STRING1   A flag which is set to 'TYPE' if only the first item *
*              in STRING2 is to be generated and to 'ALLD' if all *
*              of the items are needed.                         *
*    STRING2   A string which will be set to ('NONE', 'ELEM',   *
*              'SET '), ('NORM', 'TTYB', 'RTTC', 'LPDB'),       *
*              ('INCL', 'OMIT'), ('DETC', 'NDET'), ('WINK',     *
*              'STDY'), ('DIMM', 'MEDM', 'BRIT', 'VBRT'), ('SOLD', *
*              'DASH', 'DDSH', 'DOTS') concatenated together.   *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDDATA ENTRY(FIXED BINARY,CHARACTER(4),            *
*      CHARACTER(28));                                          *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to store and retrieve information *
*  in the auxiliary character string associated with each set or*
*  element.                                                     *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDAUXL(ISOE,STRING1,STRING2);                         *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ISOE     The identification of a set or element whose      *
*             auxiliary character string is to be stored or     *
*             retrieved.                                        *
*    STRING1  Either 'PUT' or 'GET' to indicate the operation to*
*             be performed.                                     *
*    STRING2  The auxiliary character string.  The length of this*
*             string should be the same as the value of the NAUXL*
*             argument supplied to IDOPEN.                      *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDAUXL ENTRY(FIXED BINARY,CHARACTER(3),CHARACTER(*));*
*                                                               *
*  The error codes produced by this procedure are:              *
*    1 - The item identified by ISOE could not be found.        *
*                                                               *
*****************************************************************
```

Certain elements, such as teletype input buffers and light pen drawing buffers, may be modified by the console operator. The application program may then read these elements into the SYSTEM/360. After the element has been read into the SYSTEM/360, the programmer then has IDIIOM orders in the SYSTEM/360 which must be interpreted. The procedures in this section may be used to recover the data in retrieved elements.

PROCEDURE	FUNCTION
IDXTTYB	Translate the information in a retrieved TTYB into SYSTEM/360 character code.
IDXLPDB	Translate the information in a retrieved LPDB into an array of points.

```
*************************************************************************
*                                                                       *
*  This procedure accepts a teletype input buffer element which         *
*  has been read from the IDIIOM by IDEGET and extracts the             *
*  character data from the element and converts it to SYSTEM/360        *
*  character code.                                                      *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDXTTYB(ELEMENT,STRING,IEXIT);                                *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    ELEMENT  A varying length character string which should            *
*             contain the teletype input buffer.                        *
*    STRING   A string which will have the extracted characters         *
*             stored into it.                                           *
*    IEXIT    If this is positive it represents the number of           *
*             characters put into STRING.  If it is negative, the       *
*             absolute value is the number of characters that           *
*             would not fit into STRING.                                *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDXTTYB ENTRY(CHARACTER(*) VARYING,CHARACTER(*),           *
*      FIXED BINARY);                                                   *
*                                                                       *
*************************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure accepts a light pen drawing buffer element which  *
*  has been read from the IDIIOM by IDEGET and extracts the     *
*  coordinates of the points on the curves.  A new curve is     *
*  started each time the light pen switch is closed.  In addition,  *
*  if a segment in the middle of a curve is erased, the original  *
*  curve becomes two separate curves.                           *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDXLPDB(ELEMENT,XARRAY,YARRAY,IARRAY,NARRAY,COUNT,    *
*      IEXIT);                                                  *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ELEMENT  A varying length character string which should    *
*             contain the light pen drawing buffer.             *
*    XARRAY   An array into which all of the X coordinates will be  *
*             placed.                                           *
*    YARRAY   An array into which all of the Y coordinates will be  *
*             placed.                                           *
*    IARRAY   An array into which the starting indices of the   *
*             curves in XARRAY-YARRAY will be placed.  For example  *
*             the second curve begins at (XARRAY(IARRAY(2)),    *
*             YARRAY(IARRAY(2))).                               *
*    NARRAY   An array into which the number of points in the   *
*             curves in XARRAY-YARRAY will be placed.  For example  *
*             the second curve contains NARRAY(2) points.       *
*    COUNT    The number of curves found in the element.        *
*    IEXIT    A zero if the element was scanned to completion, or  *
*             a 1, 2, 3, or 4 depending on which of XARRAY,     *
*             YARRAY, IARRAY, or NARRAY was not large enough.   *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDXLPDB ENTRY(CHARACTER(*) VARYING,(*) FIXED BINARY,  *
*      (*) FIXED BINARY,(*) FIXED BINARY,(*) FIXED BINARY,      *
*      FIXED BINARY,FIXED BINARY);                              *
*                                                               *
*****************************************************************
```

This section describes the procedures which may be used to control attentions or interrupts from the IDIIOM. These procedures and their functions are:

PROCEDURE	FUNCTION
IDEATTN	Enable interrupts for queuing.
IDSATTN	Enable interrupts for immediate processing.
IDDATTN	Disable interrupts.
IDRATTN	Obtain a record of an interrupt from the queue.

Interrupts are enabled or disabled by specifying a four character code. When an interrupt is reported, this code and additional useful information is made available. This information which describes the interrupt is contained in an attention data structure whose declaration is:

```
DECLARE 1 ATTND,
          2 STRING CHARACTER(4),
          2 ARRAY(5) FIXED BINARY;
```

The valid interrupt codes, their meaning, and the information which is made available in ARRAY are:

INTERRUPT CODES	ARRAY(1)	A DESCRIPTION OF THE INTERRUPT AND THE MEANING OF THE VALUES IN ARRAY(2)...ARRAY(5)
'NONE'	0	A possible return value when IDRATTN returns because the specified time has expired.
'GOOF'	1	The program in the 620/I has detected an error. ARRAY(2) indicates the nature of the error. Possible values are: 1 You have run out of space in the 620/I for display elements. 2 You have run out of space in the current light pen drawing buffer.
'PFKM'	2	Program function key make. ARRAY(2) gives the key number (1 through 32) which caused the interrupt. ARRAY(3) will contain a one if the key had its light on or a zero if the key had its light off. The keyboard is arranged in 4 rows of 8 keys. Key number one is in the top row on the left, key number two is in the top row next to key number one, etc.
'PFKB'	3	Program function key break. ARRAY(2) gives the key number (1 through 32) which caused the interrupt. ARRAY(3) will contain a one if the key had its light on or a zero if the key had its light off.

INTERRUPT CODES	ARRAY(1)	A DESCRIPTION OF THE INTERRUPT AND THE MEANING OF THE VALUES IN ARRAY(2)...ARRAY(5)
'PFLM'	4	This works the same as 'PFKM' except that the light for the key must be on before this interrupt is generated.
'PFLB'	5	This works the same as 'PFKB' except that the light for the key must be on before this interrupt is generated.
'LPSM'	6	Light pen switch make.
'LPSB'	7	Light pen switch break.
'LPDT'	8	Light pen detect on an element. ARRAY(2) gives the identification of the element and ARRAY(3) contains the byte offset of the order in the element corresponding to where the light pen was pointing. ARRAY(4) and ARRAY(5) give the X and Y coordinates of the beam when the interrupt occurred.
'LPCR'	9	An alternate to using 'LPSM', 'LPSB', and 'LPDT'. When 'LPCR' is enabled, the light pen works as follows: 1. A detectable element will brighten when the light pen is pointing at it. 2. When the pen is pointing at a brightened element and the light pen switch is closed, a 'LPCR' interrupt is sent to the SYSTEM/360. ARRAY(2)...ARRAY(5) are the same as for 'LPDT'. 3. Light pen switch break is ignored.
'TCMM'	10	This interrupt is generated every time the tracking cross has moved a specified distance from its initial position or its last reported position. ARRAY(4) and ARRAY(5) give the X and Y coordinates of the tracking cross.
'FRAM'	11	This interrupt is generated at the end of a specified number of display regeneration cycles.
'ITYP'	12	Generated by the escape key on the teletype or keyboard.

In certain cases, for example 'PFKM' and 'PFLM', these interrupt codes specify contradictory action. In this example 'PFLM' takes precedence over 'PFKM', that is, if both are enabled the resulting action will be that of 'PFLM'. 'PFLB' takes precedence over 'PFKB'. 'LPCR' takes precedence over 'LPSM', 'LPSB', and 'LPDT'.

When an interrupt is enabled the programmer must specify the disposition of the display after the interrupt is reported. This disposition may be either 'STOP' or 'KRUN'. If 'STOP' is specified the display will stop regenerating when the interrupt is reported. If 'KRUN' is specified the display will keep running after an interrupt is reported. The 'STOP' disposition will usually assure that the SYSTEM/360 will have answered an interrupt before another interrupt can be generated. The 'STOP' disposition will therefore limit the interrupt queue to at most one item and keep the SYSTEM/360 and 620/I in phase. The 'KRUN' disposition should be reserved for special situations where it is reasonable to have the interrupt queue grow to greater lengths. It is the programmers responsibility to make sure that interrupts such as 'LPDT' and 'FRAM' are not enabled with the 'KRUN' disposition. Such an action could flood the SYSTEM/360 with interrupts.

```
*****************************************************************
*                                                               *
*  This procedure enables interrupts for processing by procedure *
*  IDRATTN.  When an enabled interrupt occurs it will be queued. *
*  Procedure IDRATTN may then interrogate this interrupt queue to *
*  obtain the record of the oldest unreported interrupt.        *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDEATTN(STRING1,STRING2);                             *
*                                                               *
*  The parameter in the calling sequence is:                    *
*    STRING1  A character string containing any of the interrupt *
*             codes concatenated together.  Any invalid data in *
*             STRING will be ignored.                           *
*    STRING2  Either 'STOP' or 'KRUN'.  If this parameter cannot *
*             be recognized then 'STOP' is assumed.             *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDEATTN ENTRY(CHARACTER(*),CHARACTER(4));          *
*                                                               *
*****************************************************************
```



```
*****************************************************************
*                                                               *
*  This procedure enables interrupts for immediate processing by a *
*  user supplied attention procedure.  The indicated procedure is *
*  executed when the interrupt occurs, no queuing of interrupts *
*  takes place.                                                 *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDSATTN(STRING1,STRING2,PROC);                        *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    STRING1  A character string containing any of the interrupt *
*             codes concatenated together.  Any invalid data in *
*             STRING will be ignored.                           *
*    STRING2  Either 'STOP' or 'KRUN'.  If this parameter cannot *
*             be recognized then 'STOP' is assumed.             *
*    PROC     The entry point to the attention processing       *
*             procedure.  The procedure statement for PROC is:  *
*               PROC: PROCEDURE(ATTND);                         *
*             Where ATTND is an attention data structure.       *
*             Procedure PROC should terminate via a RETURN or END *
*             statement.  Other means of exiting from PROC will *
*             usually cause trouble.                            *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDSATTN ENTRY(CHARACTER(*),CHARACTER(4),ENTRY);    *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure disables interrupts.  The interrupts may have *
*  been enabled by procedures IDEATTN or IDSATTN.               *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDDATTN(STRING);                                      *
*                                                               *
*  The parameter in the calling sequence is:                    *
*    STRING    A character string containing any of the interrupt *
*              codes concatenated together.  Any invalid data in *
*              STRING will be ignored.                          *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDDATTN ENTRY(CHARACTER(*));                       *
*                                                               *
*****************************************************************
```



```
*****************************************************************
*                                                               *
*  This procedure returns information on the oldest unreported  *
*  queued interrupt.  If no interrupts have been queued the     *
*  procedure may either return immediately, wait a given time   *
*  interval for an interrupt, or wait indefinitely for an       *
*  interrupt.  An interrupt is deleted from the queue when it is *
*  reported.                                                    *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDRATTN(TIME,ATTND);                                  *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    TIME      If TIME is positive or zero, it specifies the time *
*              in milliseconds to be spent waiting for an       *
*              interrupt.  In this case control will return from *
*              this procedure as soon as an interrupt has occurred, *
*              or when the time interval has run out, whichever *
*              comes sooner.  If TIME is positive, the unexpired *
*              time (or a zero) will be placed in TIME on exit. *
*              If TIME is negative, this procedure will wait    *
*              indefinitely for an interrupt.                   *
*    ATTND     An attention data structure.                     *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDRATTN ENTRY(FIXED BINARY(31,0),1,2 CHARACTER(4),  *
*      2 (5) FIXED BINARY);                                     *
*                                                               *
*****************************************************************
```

This section describes a set of procedures which may be used to control the tracking cross. When the tracking cross is on the screen it will, under normal conditions, be displayed sixty times a second regardless of how much other material is being displayed at the same time. The tracking rate therefore is relatively independent of the amount of material being displayed. Note however that the display must be regenerating before the tracking cross will appear on the screen. These procedures and their functions are:

PROCEDURE	FUNCTION
IDTCPUT	Put the tracking cross on the screen.
IDTCRMV	Remove the tracking cross from the screen.
IDTCPOS	Read the current coordinates of the tracking cross.
IDTCDEL	Set the distance that the tracking cross may be moved before a TCMN interrupt is generated, or before a new line segment is inserted into a light pen drawing buffer.
IDTCDMD	Set the mode for a light pen drawing buffer.

Freehand figures may be drawn on the face of the CRT by having a light pen drawing buffer (LPDB) element in the IDIIOM and the tracking cross on the screen. If the IDIIOM is in the drawing mode, the console operator may draw by holding the light pen switch down while moving the tracking cross with the light pen. Releasing the light pen switch causes the current curve to be completed. Many curves may be put into one LPDB element. The minimum size of each line segment (except the last) in the curve is set by procedure IDTCDEL. If this line segment size is large a "rubber band" effect will be apparent. The LPDB element should not be in the light pen detectable state when the IDIIOM is in the drawing mode.

When the erase mode is selected, pointing to a line segment while the light pen switch is depressed will cause that segment to disappear. The space in the LPDB element is not reusable. The LPDB must be in the light pen detectable state for erasing to take place. Thus if the LPCR interrupt is enabled when the IDIIOM is in the erase mode, it is possible to get a LPCR interrupt sent to the SYSTEM/360 if the console operator first points to the LPDB element and then presses the light pen switch. The programmer should ignore this interrupt if it may cause trouble.

A LPDB element is of fixed size. The console operator may therefore completely fill this element with curves he has drawn. When this happens, a GOOF interrupt is sent to the SYSTEM/360 (if GOOF is enabled) and the IDIIOM is taken out of the light pen drawing mode and put into the inactive mode.

```
*****************************************************************
*                                                               *
*  This procedure may be used to put the tracking cross on the  *
*  screen.  The position where the tracking cross is to appear is *
*  given.  If the tracking cross is already on the screen, it is *
*  re-positioned.                                               *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDTCPUT(XTC,YTC);                                     *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    XTC       The X coordinate where the tracking cross is to  *
*              appear.                                          *
*    YTC       The Y coordinate where the tracking cross is to  *
*              appear.                                          *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDTCPUT ENTRY(FIXED BINARY,FIXED BINARY);          *
*                                                               *
*****************************************************************
```



```
*****************************************************************
*                                                               *
*  This procedure may be used to remove the tracking cross from *
*  the screen.                                                  *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDTCRMV;                                              *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDTCRMV ENTRY;                                     *
*                                                               *
*****************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure may be used to determine the position of the  *
*  tracking cross on the screen.                                *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDTCPOS(XTC,YTC);                                     *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    XTC      The current X coordinate of the tracking cross.   *
*    YTC      The current Y coordinate of the tracking cross.   *
*  If the tracking cross is not present, XTC and YTC are set to *
*  1024.                                                        *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDTCPOS ENTRY(FIXED BINARY,FIXED BINARY);          *
*                                                               *
*****************************************************************
```



```
*****************************************************************
*                                                               *
*  This procedure may be used to specify the distance that the  *
*  tracking cross may be moved before a TCHN interrupt is       *
*  generated, or before a new line segment is inserted into a   *
*  light pen drawing buffer.                                    *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDTCDEL(DELTC);                                       *
*                                                               *
*  The parameter in the calling sequence is:                    *
*    DELTC    The distance (in raster units) that the tracking  *
*             cross may be moved before the required action is  *
*             performed.  If DELTC is less than one, a value of *
*             one is assumed.                                   *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDTCDEL ENTRY(FIXED BINARY);                       *
*                                                               *
*****************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure may be used to set the mode of operation for          *
*  using a light pen drawing buffer (LPDB) element.  The mode may       *
*  be set to inactive mode (the suggested mode when no LPDB             *
*  element is present), drawing mode, and erasing mode.  Initially      *
*  the IDIIOM is in the inactive mode.                                  *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDTCDMD(STRING);                                              *
*                                                                       *
*  The parameter in the calling sequence is:                            *
*    STRING   Any of 'INAC', 'DRAW', or 'ERAS'.  If this parameter      *
*             cannot be recognized, 'INAC' is assumed.                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDTCDMD ENTRY(CHARACTER(4));                               *
*                                                                       *
*************************************************************************
```

This section describes some procedures which do not fall into any of the previous sections. These procedures and their functions are:

PROCEDURE	FUNCTION
IDRGEN	Control the regeneration of the display.
IDNFRAM	Specify the number of display regeneration cycles to be done before a FRAM interrupt is generated.
IDLITES	Control which of the program function key lights are on or off.
IDTRACE	Produce a trace-back through the element-subroutines for the last LPDT or LPCR interrupt.
IDESUER	Specify if checking for element-subroutines is to occur.
IDSHUTR	Control the camera shutter in the double pulse animation mode.
IDEPRNT	Print the contents of a display element.
IDEPCH1	Punch a single element as a declaration of a character string with an initial attribute.
IDEPCHN	Punch an array of elements as a declaration of an array of character strings with an initial attribute.
IDSYMB	Generate data which may be passed to procedure IDPTLN to create characters of arbitrary size.

PROCEDURE IDRGEN SECTION 2.7.1

```
*****************************************************************
*                                                               *
*  This procedure is used to turn display regeneration on or off.  *
*  When the display is turned on, it remains on until it is turned  *
*  off with this or another procedure, or until an interrupt with  *
*  the 'STOP' option occurs.                                    *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDRGEN(STRING);                                       *
*                                                               *
*  The parameter in the calling sequence is:                    *
*    STRING   Either 'STRT' or 'STOP'.  'STRT' is an abreviation  *
*             for start.  If this parameter cannot be recognized,  *
*             'STOP' is assumed.  In addition STRING may also be  *
*             'SWST' (for switch and start).  This is a special  *
*             options which will cause stereo elements (elements  *
*             with an identification greater than 1000 or less  *
*             than -1000) to have their INCL-OMIT state changed,  *
*             and the display then restarted.                    *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDRGEN ENTRY(CHARACTER(4));                        *
*                                                               *
*****************************************************************
```

PROCEDURE IDNFRAM SECTION 2.7.2

```
*****************************************************************
*                                                               *
*  This procedure may be used to specify the number of times the  *
*  display is regenerated before a FRAM interrupt is generated.  *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDNFRAM(NFRAM);                                       *
*                                                               *
*  The parameter in the calling sequence is:                    *
*    NFRAM    The number of times the display is regenerated    *
*             before a FRAM interrupt is generated.             *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDNFRAM(FIXED BINARY);                             *
*                                                               *
*****************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure may be used to control the lights on the program      *
*  function keyboard.  The on-off state of these lights may be          *
*  set, modified, or read.                                              *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDLITES(STRING,LITEBITS);                                     *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    STRING    Any of ('RSET', 'LAND', 'LOR ', 'XOR ', 'READ') to       *
*              indicate the operation to be performed on the            *
*              lights.  If this parameter cannot be recognized,         *
*              'RSET' is assumed.                                       *
*    LITEBITS A bit string of 32 bits.  The first bit corresponds       *
*              to light number one, the second bit corresponds to       *
*              light number two, etc.  For 'RSET', the lights           *
*              whose corresponding bit is a one is turned on and        *
*              all other lights are turned off.  For 'LAND',            *
*              'LOR ', and 'XOR ', the logical operation is             *
*              performed on the current state of the lights and         *
*              LITEBITS to obtain the new state of the lights.          *
*              For 'READ', LITEBITS is set to the current state         *
*              of the keyboard lights.                                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDLITES ENTRY(CHARACTER(4),BIT(32));                       *
*                                                                       *
*************************************************************************
```

```
*****************************************************************
*                                                               *
*  When an LPDT or LPCR interrupt occurs, the information that is  *
*  supplied to the programmer consists of data concerning the   *
*  element that was detected.  If the element was acting as an  *
*  element-subroutine which is being displayed more than once, the *
*  interrupt information may not be enough to identify the      *
*  detected image.  This procedure allows the programmer to obtain *
*  a trace-back showing the identification of the elements which *
*  resulted in the execution of the detected element.  The trace-  *
*  back is supplied for the most recent LPDT or LPCR interrupt.  *
*  The programmer must not have transmitted or deleted any      *
*  elements between the time that the interrupt occurred and the *
*  time that this procedure is called.  The trace-back is limited  *
*  to the first four levels.                                    *
*                                                               *
*  The calling sequence is;                                     *
*    CALL IDTRACE(TRACTAB);                                     *
*                                                               *
*  The parameter in the calling sequence is:                    *
*    TRACTAB  A array which contains the trace information.     *
*             TRACTAB(1) contains the number of elements in the *
*             trace-back.  If the last detect was on an element *
*             which was not an element-subroutine, then TRACTAB(1) *
*             is set to zero.  If TRACTAB(1) is greater than zero, *
*             then (TRACTAB(2),TRACTAB(3)) contain the identif- *
*             ication of the highest level executing element, and *
*             the byte offset of the calling sequence respectively. *
*             The rest of TRACTAB contains information similar to *
*             that in (TRACTAB(2),TRACTAB(3)).                  *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDTRACE ENTRY((9) FIXED BINARY);                   *
*                                                               *
*****************************************************************
```

This procedure is used to supply the package with information on the presence or absence of element-subroutines. Normally the package does not expect element-subroutines and does not check for them. A substantial amount of time is saved in the IDIIOM by avoiding this checking when an element is being added, deleted, or changed. The extra time consumed in checking for element-subroutines is usually noticable only when some form of animation is being attempted.

The calling sequence is:

```
CALL IDESUBR(STRING);
```

The parameter in the calling sequence is:

STRING — Either 'ON' or 'OFF' to indicate that checking is to be turned on or off. When checking is turned off, no element-subroutines may be present in the IDIIOM and no element-subroutines may be transmitted.

The entry declaration for this procedure is:

```
DECLARE IDESUBR ENTRY(CHARACTER(*));
```


This procedure may be used to open and close the shutter on the movie camera when the 620/I is in the double pulse animation mode. In all other modes a call to this procedure will be ignored.

The calling sequence is:

```
CALL IDSHUTR(STRING);
```

The parameter in the calling sequence is:

STRING — Either 'OPEN' or 'CLOS' to indicate if the shutter is to be opened or closed.

The entry declaration for this procedure is:

```
DECLARE IDSHUTR ENTRY(CHARACTER(4));
```

PROCEDURE IDEPRNT SECTION 2.7.7

```
*****************************************************************
*                                                               *
*  This procedure will print the current contents of a varying  *
*  character string in binary, octal, and hexadecimal.          *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDEPRNT(ELEMENT);                                     *
*                                                               *
*  The parameters in the calling sequence are:                  *
*    ELEMENT  A varying character string which will be printed. *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDEPRNT ENTRY(CHARACTER(*) VARYING);               *
*                                                               *
*****************************************************************
```

PROCEDURES IDEPCH1 AND IDEPCHN SECTION 2.7.8

```
*****************************************************************
*                                                               *
*  These procedures can be used to punch out cards which contain*
*  an element or elements in a form in which they can easily be *
*  incorporated into another PL/1 program.  Procedure IDEPCH1 will*
*  generate card images in the form:                            *
*    DECLARE XXXX CHARACTER(NN) STATIC VARYING INITIAL('...');  *
*  while IDEPCHN generates card images in the form:             *
*    DECLARE XXXX(MM) CHARACTER(NN) STATIC VARYING INITIAL('...',*
*      '...',...);                                              *
*  The purpose of this procedure is to enable the programmer to *
*  generate static elements in a compact form and incorporate them*
*  into a program in such a manner that they are not re-created *
*  each time the program executes.  The card images are put into a*
*  data set with a DD name of PUNCH.                            *
*                                                               *
*  The calling sequences are:                                   *
*    CALL IDEPCH1(STRING,ELEMENT);                              *
*    CALL IDEPCHN(STRING,ELEMENTS);                             *
*                                                               *
*  The parameters in the calling sequences are:                 *
*    STRING     A character string containing the name of the   *
*               character string in the declaration being created.*
*    ELEMENT    A varying character string containing the element.*
*    ELEMENTS   An array of varying character strings containing*
*               the elements.                                   *
*                                                               *
*  The entry declarations for these procedures are:             *
*    DECLARE IDEPCH1 ENTRY(CHARACTER(*),CHARACTER(*) VARYING);  *
*    DECLARE IDEPCHN ENTRY(CHARACTER(*),(*) CHARACTER(*) VARYING);*
*                                                               *
*****************************************************************
```

```
*****************************************************************************
*                                                                           *
*  This procedure may be used to generate characters of arbitrary           *
*  size for the IDIIOM (or any other display or plotting device             *
*  which can generate straight line segments on a raster grid). The         *
*  input to this procedure is a character string containing the             *
*  characters to be converted to a stroke representation. The               *
*  output is an X and Y array of coordinates representing the end           *
*  points of concatenated line segments , and a bit string which           *
*  contains the blanking bits for the line segments. The first              *
*  point in the X and Y array is always (0,0). The height of the            *
*  characters is from 0 to 6 units, although some lower case letters        *
*  extend down to -2 units. The width of the characters varies from         *
*  2 to 4 units with one unit between characters. The last point in         *
*  the X and Y arrays is always (N,0) where (N-1) is the greatest Y         *
*  coordinate of any line segment in the characters ((N,0) is the           *
*  point where the next character would start). The last line               *
*  segment is always blanked. It is the programmers responsibility          *
*  to scale and translate the points in the X and Y arrays to raster        *
*  unit coordinates. Note that instead of simple scaling and                *
*  translating, the programmer could use a sheer transformation to          *
*  generate italics or a projective transformation to generate              *
*  perspective views of the letters. After the transformation is            *
*  complete, the X and Y arrays and the blanking bits are ready to          *
*  be used as input to procedure IDPTLN.                                    *
*                                                                           *
*  The calling sequence is:                                                 *
*    CALL IDSYMB(STRING,XARRAY,YARRAY,BLKBITS,NARRAY);                      *
*                                                                           *
*  The parameters in the calling sequence are:                              *
*    STRING   A character string containing the characters to be            *
*             converted to stroke representation. Any valid                 *
*             SYSTEM/360 character is permitted.                            *
*    XARRAY   An array which will have the X coordinates of the line        *
*             segments placed in it.                                        *
*    YARRAY   An array which will have the Y coordinates of the line        *
*             segments placed in it.                                        *
*    BLKBITS  A bit string which will have the blanking bits placed         *
*             in it.                                                        *
*    NARRAY   The number of points placed in XARRAY and YARRAY. The         *
*             number of blanking bits put into BLKBITS is equal to          *
*             (NARRAY-1). If XARRAY, YARRAY, or BLKBITS was not             *
*             large enough to contain all of the generated data,            *
*             then NARRAY is set to minus one.                              *
*                                                                           *
*  The entry declaration for this procedure is:                             *
*    DECLARE IDSYMB(CHARACTER(*),(*) FIXED BINARY,(*) FIXED BINARY,         *
*      BIT(*),FIXED BINARY);                                                *
*                                                                           *
*****************************************************************************
```

This section describes procedures which are not normally used by a programmer using this package. These procedures are called by the procedures described in the preceding sections. The procedures and their functions are:

PROCEDURE	FUNCTION
IDERRP	Report errors to the user of these procedures.
IDERRI	Obtain information on errors reported by the procedures at execution time.
IDREAD	Read a record from the link.
IDWRITE	Write a record to the link.
IDWRIT2	Write a record to the link.
IDEOPR	End of operation procedure for link I/O transmission.
IDATTN	Attention processing procedure for the link.
IDCBENT	Search for, or create an entry in the element control block.
IDCBGET	Obtain information from a set connector.
IDCBPUT	Store information into a set connector.
LKOPEN	Open the DCB for the link.
LKREAD	Initiate data transmission from the link.
LKWRITE	Initiate data transmission to the link.
LKCLOSE	Close the DCB for the link.
LKDATTN	Check for and process a deferred attention.
ECBCTRL	Exercise RESET, POST, and WAIT control over an event control block.
TIMCTRL	Exercise control over the interval timer.
AXSAVE	Save the entry point to a procedure for execution at a later time.
AXCALL	Execute a procedure whose entry point has been previously saved.
EBTOID	Translate EBCDIC character codes to IDIIOM character codes.
IDTOEB	Translate IDIIOM character codes to EBCDIC character codes.
VCONC	Concatenate a given character string onto a varying character string.

```
*************************************************************************
*                                                                       *
*  These two procedures are used to control errors detected by the      *
*  procedures in the IDIIOM scope package.  When an error is            *
*  detected, procedure IDERRP is called.  IDERRP saves the              *
*  arguments to it and raises the IDERROR condition.  If there is       *
*  no ON UNIT for this condition, or if the ON UNIT gives a             *
*  normal return, then an error message is printed and a dump is        *
*  produced.  Procedure IDERRI may be called at any time to             *
*  obtain the arguments supplied to IDERRP the last time it was         *
*  called.                                                              *
*                                                                       *
*  The calling sequences are:                                           *
*    CALL IDERRP(STRING,INDEX);                                         *
*    CALL IDERRI(STRING,INDEX);                                         *
*                                                                       *
*  The parameters in the calling sequences are:                         *
*    STRING    A character string which contains the name of the        *
*              procedure which found the error.                         *
*    INDEX     An integer which is the error code.                      *
*                                                                       *
*                                                                       *
*  The entry declarations for these procedures are:                     *
*    DECLARE IDERRP ENTRY(CHARACTER(*),FIXED BINARY);                   *
*    DECLARE IDERRI ENTRY(CHARACTER(8),FIXED BINARY);                   *
*                                                                       *
*************************************************************************
```

```
*************************************************************************
*                                                                       *
*  These procedures may be used to read and write fixed length          *
*  records on the link.  Error conditions are raised if an error        *
*  occurs in transmission or if the read-write count does not           *
*  match the count on the link.                                         *
*                                                                       *
*  The calling sequences are:                                           *
*    CALL IDREAD(STRING);                                               *
*    CALL IDWRITE(STRING);                                              *
*    CALL IDWRIT2(STRING,STRING2);                                      *
*                                                                       *
*  The parameters in the calling sequences are input or output          *
*  buffers.  In the case of IDWRIT2, the two buffers are                *
*  transmitted as a single record.                                      *
*                                                                       *
*  The entry declarations for these procedures are:                     *
*    DECLARE IDREAD ENTRY(CHARACTER(*));                                *
*    DECLARE IDWRITE ENTRY(CHARACTER(*));                               *
*    DECLARE IDWRIT2 ENTRY(CHARACTER(*),CHARACTER(*));                  *
*                                                                       *
*  The error codes produced by procedure IDREAD are:                    *
*    1 - A transmission error has occurred.                             *
*    2 - Transmission was complete but a residual byte count            *
*        remains.                                                       *
*  The error codes produced by procedure IDWRITE or IDWRIT2 are:        *
*    1 - A transmission error has occurred.                             *
*    2 - Data has been lost in transmission.                            *
*                                                                       *
*************************************************************************
```

```
*****************************************************************
*                                                               *
*  This procedure is the end of operation procedure for link I/O *
*  transmission.  It is called by the system to signal end of    *
*  operation.                                                   *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDEOPR(IEOPR,IRDWT);                                  *
*                                                               *
*  The values of the parameters in the calling sequence have the *
*  following meaning.                                           *
*    IEOPR=-3  A transmission error has occurred.               *
*    IEOPR=-2  Data has been lost in transmission (write only). *
*    IEOPR=-1  A 2 second interval has passed.  During this time *
*              the link did not become ready to initiate        *
*              transmission.                                    *
*    IEOPR=0   Transmission was complete.                       *
*    IEOPR>0   Transmission was complete but a residual read    *
*              byte count remains.  The value of IEOPR is this  *
*              count.                                           *
*    IRDWT=1   This call is signalling the end of a write       *
*              operation.                                       *
*    IRDWT=2   This call is signalling the end of a read        *
*              operation.                                       *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDEOPR ENTRY(FIXED BINARY,FIXED BINARY);           *
*                                                               *
*****************************************************************
```



```
*****************************************************************
*                                                               *
*  This procedure is the attention processing procedure for the *
*  link.  It is called by the system to signal that an attention *
*  has been received from the link.                             *
*                                                               *
*  The calling sequence is:                                     *
*    CALL IDATTN;                                               *
*                                                               *
*  When an attention is recieved from the link, this procedure  *
*  reads an attention record.  The record is processed and the  *
*  information is either queued for IDRATTN or the procedure    *
*  specified by IDSATTN is called.                              *
*                                                               *
*  The entry declaration for this procedure is:                 *
*    DECLARE IDATTN ENTRY;                                      *
*                                                               *
*****************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure may be used to search the element control block       *
*  for a block of a given type with a given identification.  If no      *
*  such block is found, a new block of that type is assigned and        *
*  it is linked into the master set.                                    *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL IDCBENT(ID,TYPE,IEXIT);                                       *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    ID        The identification of the block to be searched for.      *
*    TYPE      The type flag of the block to be searched for            *
*              ('E000' for element, 'S000' for set).                    *
*    IEXIT     An exit flag which is set to zero if a block has         *
*              been found or assigned, a one if a block with the        *
*              given identification but the wrong type was found,       *
*              and a two if there were no more blocks available to      *
*              assign.                                                  *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE IDCBENT ENTRY(FIXED BINARY,CHARACTER(4),                   *
*      FIXED BINARY);                                                   *
*                                                                       *
*************************************************************************
```



```
*************************************************************************
*                                                                       *
*  These procedures may be used to get(put) information from(into)      *
*  the set connectors of the element control block.                     *
*                                                                       *
*  The calling sequences are:                                           *
*    CALL IDCBGET(IENT,ISET,ENTVAL,SETVAL);                             *
*    CALL IDCBPUT(IENT,ISET,ENTVAL,SETVAL);                             *
*                                                                       *
*  The parameters in the calling sequences are:                         *
*    IENT,ISET       Indicates the ISET-th set connector of the         *
*                    IENT-th entry.                                     *
*    ENTVAL,SETVAL   The contents of the set connectors.                *
*                                                                       *
*  The entry declarations for these procedures are:                     *
*    DECLARE IDCBGET ENTRY(FIXED BINARY,FIXED BINARY,FIXED BINARY,      *
*      FIXED BINARY);                                                   *
*    DECLARE IDCBPUT ENTRY(FIXED BINARY,FIXED BINARY,FIXED BINARY,      *
*      FIXED BINARY);                                                   *
*                                                                       *
*************************************************************************
```

```
*****************************************************************************
*                                                                           *
*  These procedures are the PL/1 versions of the basic link I/O             *
*  routines.  Procedure LKOPEN opens the link DCB and supplies the          *
*  entry points to an attention-processing procedure and an end-of-         *
*  operation procedure.  The attention-processing procedure will be         *
*  entered whenever the link gives an interrupt signal.  Procedure          *
*  LKREAD will attempt to read a record from the link into a                *
*  character string.  Procedure LKWRITE will attempt to write a             *
*  record to the link from a character string.  Procedures LKREAD           *
*  and LKWRITE attempt to initiate data transmission.  The termin-          *
*  ation of this attempt is signaled by the execution of the end-           *
*  of-operation procedure.  Procedure LKCLOSE will close the DCB.           *
*  Procedure LKDATTN will check to see if an attention has been             *
*  deferred during an I/O operation.  The problem here is that              *
*  OS/360 may get the proper order of an end-of-operation and an            *
*  attention mixed up.                                                      *
*                                                                           *
*  The calling sequences are:                                               *
*    CALL LKOPEN(ATTN,EOPR);                                                *
*    CALL LKREAD(STRING);                                                   *
*    CALL LKWRITE(STRING,STRING2);                                          *
*    CALL LKCLOSE;                                                          *
*    CALL LKDATTN;                                                          *
*                                                                           *
*  The entry declarations for these procedures are:                         *
*    DECLARE LKOPEN ENTRY(ENTRY,ENTRY);                                     *
*    DECLARE LKREAD ENTRY(CHARACTER(*));                                    *
*    DECLARE LKWRITE ENTRY(CHARACTER(*),CHARACTER(*));                      *
*    DECLARE LKCLOSE ENTRY;                                                 *
*    DECLARE LKDATTN ENTRY;                                                 *
*                                                                           *
*  The procedure statements for ATTN and EOPR are:                          *
*    ATTN: PROCEDURE;                                                       *
*    EOPR: PROCEDURE(IEOPR,IRDWT);                                          *
*  The parameters IEOPR and IRDWT are declared by:                          *
*    DECLARE (IEOPR,IRDWT) FIXED BINARY(31,0);                              *
*  The value of IEOPR has the following meaning:                            *
*    IEOPR=-3  A transmission error has occurred.                           *
*    IEOPR=-2  Data has been lost in transmission (LKWRITE only).           *
*    IEOPR=-1  A 2 second interval has passed.  During this time            *
*              the link did not become ready to initiate                    *
*              transmission.                                                *
*    IEOPR=0   Transmission was complete.                                   *
*    IEOPR>0   Transmission was complete but a residual read                *
*              byte count remains.  The value of IEOPR is this              *
*              count.                                                       *
*  The value of IRDWT has the following meaning:                            *
*    IRDWT=1   This call is signaling the end of a write operation.         *
*    IRDWT=2   This call is signaling the end of a read operation.          *
*                                                                           *
*****************************************************************************
```

```
*************************************************************************
*                                                                       *
*  This procedure may be used to control an event control block         *
*  (ECB).  This enables a PL/1 programmer to synchronize the            *
*  operation of a program with attentions being generated by devices    *
*  not supported by OS/360.                                             *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL ECBCTRL(ECB,STRING);                                          *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    ECB       The event control block to be acted upon.                *
*    STRING    Any of 'RSET', 'WAIT', or 'POST'.                        *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE ECBCTRL ENTRY(FIXED BINARY(31,0),CHARACTER(4));            *
*                                                                       *
*************************************************************************
```



```
*************************************************************************
*                                                                       *
*  This procedure may be used to control the interval timer.  The       *
*  timer may be set with a time interval, and later, the amount of      *
*  time remaining may be determined.                                    *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL TIMCTRL(STRING,FLAG,TIME);                                    *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    STRING    Either 'STIM' or 'TTIM'.  When 'STIM' is given, the      *
*              timer is set to the value in TIME, and FLAG specifies    *
*              an ECB which will be posted when the time interval       *
*              expires.  When 'TTIM' is given, the remaining time       *
*              is placed in TIME, and the time interval is cancelled    *
*              if FLAG is zero.                                         *
*    FLAG      An ECB or flag as described above.                       *
*    TIME      A time interval which is always given in milliseconds.   *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE TIMCTRL ENTRY(CHARACTER(4),FIXED BINARY(31,0),             *
*      FIXED BINARY(31,0));                                             *
*                                                                       *
*************************************************************************
```

PROCEDURES AXSAVE AND AXCALL — SECTION 2.8.10

These procedures may be used by the PL/1 programmer to select a procedure for execution in one part of a program, and actually carry out the calling of the procedure in another part of the program. Procedure AXSAVE will save the entry point supplied to it, and procedure AXCALL will use this saved entry point to call the procedure. The remaining parameters in procedure AXCALL are passed to the procedure that is to be called.

The calling sequences are:

```
CALL AXSAVE(ENTRY,SAVELOC);
CALL AXCALL(SAVELOC,...);
```

The entry declarations for these procedures are:

```
DECLARE AXSAVE(ENTRY,FIXED BINARY(31,0));
DECLARE AXCALL(FIXED BINARY(31,0),...);
```

PROCEDURES EBTOID AND IDTOEB — SECTION 2.8.11

These procedures are used to translate between SYSTEM/360 (EBCDIC) and IDIIOM character codes. Procedure EBTOID translates from EBCDIC to IDIIOM, and procedure IDTOEB translates from IDIIOM to EBCDIC. Characters which cannot be translated from one representation to another are translated to blanks. The only exception to this rule is that lower case EBCDIC characters are translated to upper case IDIIOM.

The calling sequences are:

```
CALL EBTOID(STRING);
CALL IDTOEB(STRING);
```

The parameter in the calling sequence is the character string to be translated.

The entry declarations for these procedures are:

```
DECLARE EBTOID ENTRY(CHARACTER(*));
DECLARE IDTOEB ENTRY(CHARACTER(*));
```

```
*************************************************************************
*                                                                       *
*  This procedure is used to concatenate a given character string       *
*  onto the current contents of a varying character string.             *
*                                                                       *
*  The calling sequence is:                                             *
*    CALL VCONC(VSTRING,STRING);                                        *
*                                                                       *
*  The parameters in the calling sequence are:                          *
*    VSTRING  A varying character string which will have STRING         *
*             concatenated onto it.                                     *
*    STRING   A character string.                                       *
*                                                                       *
*  Note that this procedure does exactly the same thing as the PL/1     *
*  statement:                                                           *
*    VSTRING=VSTRING||STRING;                                           *
*  The problem is that, at the time this procedure was written, PL/1    *
*  generated code which was unacceptable in its extravagant use of      *
*  core space.  In particular, a temporary VDA was generated to hold    *
*  the concatenated string.                                             *
*                                                                       *
*  The entry declaration for this procedure is:                         *
*    DECLARE VCONC ENTRY(CHARACTER(*) VARYING,CHARACTER(*));            *
*                                                                       *
*************************************************************************
```

```
EXAMPLE: PROCEDURE OPTIONS(MAIN);

/*  THE INITIAL DISPLAY CREATED BY THIS PROGRAM CONSISTS OF SOME
    DESCRIPTIVE TEXT, SOME ACTION MESSAGES WHICH MAY BE SELECTED
    WITH THE LIGHT PEN, A GRID, AND THE TRACKING CROSS.  THE IDIOM
    OPERATOR MAY USE THE TRACKING CROSS TO DRAW A SIMPLE CLOSED
    CURVE ON THE GRID.  WHEN THE CURVE HAS BEEN DRAWN, THE OPERATOR
    MAY USE ONE OF THE ACTION MESSAGES TO CAUSE THE ENCLOSED AREA
    AND ARC LENGTH OF THE CURVE TO BE COMPUTED.  AT ALL TIMES THERE
    WILL BE ACTION MESSAGES ON THE SCREEN WHICH MAY BE USED TO
    TERMINATE THE PROGRAM OR CLEAR THE SCREEN.  THE COMPUTED AREA
    WILL BE MEANINGLESS IF THE CURVE IS NOT A SIMPLE CLOSED CURVE.  */

DECLARE IDOPEN ENTRY(FIXED BINARY,FIXED BINARY,FIXED BINARY);
DECLARE IDCLOSE ENTRY;
DECLARE IDPOS ENTRY(CHARACTER(4),FIXED BINARY,FIXED BINARY,
  CHARACTER(*) VARYING,FIXED BINARY);
DECLARE IDTEXT ENTRY(CHARACTER(*),CHARACTER(*),CHARACTER(*) VARYING,
  FIXED BINARY);
DECLARE IDPTLN ENTRY(CHARACTER(*),(*) FIXED BINARY,(*) FIXED BINARY,
  FIXED BINARY,BIT(*),CHARACTER(*) VARYING,FIXED BINARY);
DECLARE IDDBUF ENTRY(FIXED BINARY,CHARACTER(*) VARYING,FIXED BINARY);
DECLARE IDEPUT ENTRY(FIXED BINARY,CHARACTER(*),CHARACTER(*) VARYING);
DECLARE IDEGET ENTRY(FIXED BINARY,CHARACTER(*) VARYING,FIXED BINARY,
  FIXED BINARY);
DECLARE IDSET ENTRY(FIXED BINARY);
DECLARE IDMODX ENTRY(FIXED BINARY,CHARACTER(*));
DECLARE IDLINK ENTRY(FIXED BINARY,FIXED BINARY,FIXED BINARY);
DECLARE IDAPPLY ENTRY(FIXED BINARY,FIXED BINARY,ENTRY,);
DECLARE IDXLPDB ENTRY(CHARACTER(*) VARYING,(*) FIXED BINARY,
  (*) FIXED BINARY,(*) FIXED BINARY,(*) FIXED BINARY,FIXED BINARY,
  FIXED BINARY);
DECLARE IDEATTN ENTRY(CHARACTER(*),CHARACTER(4));
DECLARE IDRATTN ENTRY(FIXED BINARY(31,0),1,2 CHARACTER(4),
  2 (5) FIXED BINARY);
DECLARE IDTCPUT ENTRY(FIXED BINARY,FIXED BINARY);
DECLARE IDTCRMV ENTRY;
DECLARE IDTCDEL ENTRY(FIXED BINARY);
DECLARE IDTCDMD ENTRY(CHARACTER(4));
DECLARE IDRGEN ENTRY(CHARACTER(4));

DECLARE XGRID(24) FIXED BINARY STATIC INITIAL(262,762,762,262,262,
      762,762,262,262,762,762,262,262,262,362,362,462,462,562,562,
      662,662,762,762);
DECLARE YGRID(24) FIXED BINARY STATIC INITIAL(200,200,300,300,400,
      400,500,500,600,600,700,700,700,200,200,700,700,200,200,700,
      700,200,200,700);
DECLARE 1 ATTND,
          2 STRING CHARACTER(4),
          2 ARRAY(5) FIXED BINARY;
```

```
DECLARE ELEMENT CHARACTER(1000) VARYING;
DECLARE (XARRAY,YARRAY)(500) FIXED BINARY;
DECLARE (IARRAY,MARRAY)(10) FIXED BINARY;
DECLARE ST35 CHARACTER(35);
DECLARE (AREA,ARCL,X1,Y1,X2,Y2) FLOAT BINARY;
DECLARE (NCURV,I,J,K) FIXED BINARY;

/*  FIRST THE DISPLAY IS INITIALIZED, THE PEN TRACKING PARAMETERS
    ARE SET, THE TITLES AND GRID ARE PUT ON THE SCREEN, THE ACTION
    MESSAGES (ELEMENTS 101, 102, AND 103) ARE PUT ON THE SCREEN,
    AND THE LPCR AND GOOF INTERRUPTS ARE ENABLED.  */
      CALL IDOPEN(10,1,0);  /*  INITIALIZE THE DISPLAY.  */
      CALL IDTCDEL(15);  /*  SET PEN TRACKING PARAMETERS.  */
      CALL IDTCDMD('DRAW');
      ELEMENT='';  /*  GENERATE TITLES AND GRID.  */
      CALL IDPOS('ABSL',120,1000,ELEMENT,I);
      CALL IDTEXT('VLRG','AREA-ARC LENGTH CALCULATOR',ELEMENT,I);
      CALL IDPOS('ABSL',208,875,ELEMENT,I);
      CALL IDTEXT('MEDM','SELECT ONE OF THE FOLLOWING...',ELEMENT,I);
      CALL IDPOS('ABSL',XGRID(1),YGRID(1),ELEMENT,I);
      CALL IDPTLN('LINECOMP',XGRID,YGRID,24,'10'B,ELEMENT,I);
      CALL IDEPUT(10,'BRIT',ELEMENT);
      ELEMENT='';  /*  GENERATE ELEMENT 101.  */
      CALL IDPOS('ABSL',236,840,ELEMENT,I);
      CALL IDTEXT('MEDM','TERMINATE PROGRAM',ELEMENT,I);
      CALL IDEPUT(101,'BRITDETC',ELEMENT);
      ELEMENT='';  /*  GENERATE ELEMENT 102.  */
      CALL IDPOS('ABSL',236,805,ELEMENT,I);
      CALL IDTEXT('MEDM','CLEAR SCREEN',ELEMENT,I);
      CALL IDEPUT(102,'BRITDETC',ELEMENT);
      ELEMENT='';  /*  GENERATE ELEMENT 103.  */
      CALL IDPOS('ABSL',236,770,ELEMENT,I);
      CALL IDTEXT('MEDM','COMPUTE AREA AND ARC LENGTH',ELEMENT,I);
      CALL IDEPUT(103,'BRITDETC',ELEMENT);
      CALL IDEATTN('LPCRGOOF','STOP');  /*  ENABLE INTERRUPTS.  */

/*  CREATE SET 500 (IT WILL HOLD THE TEMPORARY PART OF THE DISPLAY),
    GENERATE A LPDB AND PUT IT ON SET 500, AND PUT THE TRACKING
    CROSS ON THE SCREEN.  */
INIT: CALL IDSET(500);  /*  CREATE SET 500.  */
      ELEMENT='';  /*  GENERATE A 1000 BYTE LPDB.  */
      CALL IDDBUF(1000,ELEMENT,I);
      CALL IDEPUT(50,'LPDBBRIT',ELEMENT);
      CALL IDLINK(50,0,500);
      CALL IDTCPUT(100,450);  /*  PUT TRACKING CROSS ON SCREEN.  */

/*  START THE DISPLAY AND WAIT FOR AN INTERRUPT.  THE INTERRUPT MAY
    BE EITHER A GOOF (LPDB FULL) OR A LPCR (ACTION MESSAGE).  */
STRT: CALL IDRGEN('STRT');
      CALL IDRATTN(-1,ATTND);
```

```
    IF ATTND.STRING='GOOF' THEN DO;
      ELEMENT='';  /*  PUT ERROR MESSAGE ON SCREEN.  */
      CALL IDPOS('ABSL',322,100,ELEMENT,I);
      CALL IDTEXT('MEDM','CURVE IS TOO LONG-LPDB FULL',ELEMENT,I);
      CALL IDEPUT(51,'WINKBRIT',ELEMENT);
      CALL IDLINK(51,0,500);
      CALL IDTCDMD('DRAW');  /*  RESTORE DRAW MODE.  */
      GO TO RETN;
    END;
    IF ATTND.ARRAY(2)=101 THEN DO;
      CALL IDCLOSE;  /*  TERMINATE THE DISPLAY.  */
      RETURN;
    END;
    IF ATTND.ARRAY(2)=102 THEN DO;
      CALL IDAPPLY(500,-1,IDMODX,'DELT');  /*  DELETE SET 500.  */
      CALL IDMODX(103,'INCL');  /*  MAKE SURE 103 IS ON SCREEN.  */
      GO TO INIT;
    END;

/*  THE AREA AND ARC LENGTH MUST NOW BE COMPUTED.  READ THE LPDB
    FROM THE IDIIOM, TRANSLATE IT TO COORDINATE DATA, AND COMPUTE
    AND DISPLAY THE AREA AND ARC LENGTH.  IN THE AREA AND ARC
    LENGTH COMPUTATION, ONE UNIT IS EQUIVALENT TO ONE HUNDRED
    RASTER UNITS.  */
    CALL IDEGET(50,ELEMENT,I,J);  /*  READ AND TRANSLATE ELT.  */
    CALL IDXLPDB(ELEMENT,XARRAY,YARRAY,IARRAY,NARRAY,NCURV,I);
    IF I¬=0 THEN DO;
      ELEMENT='';  /*  PUT ERROR MESSAGE ON SCREEN.  */
      CALL IDPOS('ABSL',336,100,ELEMENT,I);
      CALL IDTEXT('MEDM','CURVE CANNOT BE UNSCRAMBLED',ELEMENT,I);
      CALL IDEPUT(52,'WINKBRIT',ELEMENT);
      CALL IDLINK(52,0,500);
      GO TO RETN;
    END;
    AREA,ARCL=0;  /*  COMPUTE AREA AND ARC LENGTH.  */
    DO I=1 TO NCURV; J=IARRAY(I);
      X1=XARRAY(J); X1=(X1-512)/100;
      Y1=YARRAY(J); Y1=(Y1-450)/100;
      DO K=J+1 TO NARRAY(I)+J-1;
        X2=XARRAY(K); X2=(X2-512)/100;
        Y2=YARRAY(K); Y2=(Y2-450)/100;
        AREA=AREA+X1*Y2-X2*Y1;
        ARCL=ARCL+SQRT((X1-X2)**2+(Y1-Y2)**2);
        X1=X2; Y1=Y2;
      END;
    END;
```

```
      ELEMENT='';  /*  PUT COMPUTED RESULTS ON SCREEN.  */
      CALL IDPOS('ABSL',266,100,ELEMENT,I);
      PUT STRING(ST35) EDIT('AREA=',ABS(AREA)/2,'ARC LENGTH=',ARCL)
        (A,F(7,2),X(5),A,F(7,2));
      CALL IDTEXT('MEDM',ST35,ELEMENT,I);
      CALL IDEPUT(53,'BRIT',ELEMENT);
      CALL IDLINK(53,0,500);

/*  EITHER THE COMPUTED RESULTS OR AN ERROR MESSAGE IS NOW ON THE
    SCREEN.  REMOVE THE TRACKING CROSS AND OMIT ELEMENT 103.  */
RETN: CALL IDTCRMV;  /*  REMOVE TRACKING CROSS.  */
      CALL IDMODX(103,'OMIT');  /*  OMIT ELEMENT 103.  */
      GO TO STRT;

END EXAMPLE;
```

This section gives the length of each procedure in the package and shows which procedures each procedure calls. This information can be used by a programmer to create efficient overlay structures.

The following table contains a line of information for each procedure in the package. Each line consists of the name of the procedure, its length, the control blocks referenced by the procedure, and the procedures called by the given procedure. The purpose and format of the control blocks is given in the next section. Thus the table shows that procedure IDOPEN is 740 (HEX) bytes in length, references the control blocks IDELCB IDATCB and IDIOCB, and calls procedures IDERRP IDWRITE and LKOPEN. Some procedures contain secondary entry points. These are indicated by having a dash in the length column of the secondary entries. Thus procedures IDSATTN and IDDATTN are secondary entries to procedure IDEATTN.

In creating overlay structures there is one restriction which must be followed. The module containing procedures LKOPEN, LKREAD, LKWRITE, LKCLOSE, and LKDATTN must not be overlain between the time that procedures IDOPEN and IDCLOSE are called. The reason for this restriction is that this module contains basic information including the first level interrupt and end-of-operation procedures. These procedures are called asynchronously (by actions initiated at the IDIIOM) and must be present in the SYSTEM/360 memory when needed.

		CONTROL BLOCKS USED				PROCEDURES CALLED																		
		IDEL	IDAT	IDIO	IOQA	IDIS	IDER	IDWR	IDCL	IDIO	IDWR	LKOP	LKRE	LKWR	LKCL	LKDA	ERPR	TAE2	ATTN	ATTN	EXEL	IDBU	IDSC	IDOR
PROCEDURE	LENGTH (HEX)	CB	CB	CB	TCB		RP	ITE	OSE	CB	ITE	EN	AD	ITE	OSE	TTN					T	F	R	DER
IDOPEN	740	X	X	X			X				X													
IDCLOSE	198	X	X	X	X		X								X									
IDPOS	3B8																							X
IDTEXT	3E0																			X				X
IDPTLN	820																							X
IDEXELT	220																							X
IDDBUF	1C0																							X
IDSCR	428																							X
IDORDER	510																							X

PROCEDURE	LENGTH (HEX)	CONTROL BLOCKS USED				PROCEDURES CALLED																				
		IDENTLIB	IDATABLK	IDINTBLK	IQUEUEDT	IDENTSEG	IDENTREC	IDENTGET	IDENTPUT	IDLINKSG	IDLINKRC	IDLINKAL	IDLINKPR	LDLINKER	LDLINKRC	LDLINKAL	LDLINKCL	LDLINKDL	ETRACEBK	ETRACEIN	ATRACEXT	AERRORXT	ERRORBK	IDENTLIB	IDENTSEG	VDENTREC
IDEPUT	580							X	X																	
IDEGET	290									X	X															
IDSET	138								X					X												
IDMODX	5B8	X						X			X				X	X										
IDLINK	380	X							X						X	X										
IDAPPLY	448	X													X											
IDMEMB	230	X							X						X											
IDDATA	3B8	X								X	X															
IDAUXL	2C0	X							X																	
IDXTTYB	198																									X
IDXLPDB	B28																									
IDEATTN	2E0		X								X															
IDSATTN	-		X								X											X				
IDDATTN	-		X								X															
IDRATTN	2C8		X		X																X	X				
IDTCPUT	328										X															
IDTCRMV	-										X															
IDTCPOS	-									X	X															
IDTCDEL	-										X															
IDTCDMD	-										X															
IDRGEN	168										X															
IDNPRAM	108										X															
IDLITES	280									X	X															
IDTRACE	158									X	X															
IDESUBR	138										X															
IDSHUTR	128										X															
IDEPRNT	850																									
IDEPCH1	DA0																									
IDEPCHN	-																									
IDSYMB	608																									

PROCEDURE	LENGTH (HEX)	CONTROL BLOCKS USED				PROCEDURES CALLED																			
		IDECB	IDATN	IDEOB	QTNDE	IDSEC	IDREP	IDERR	IDREA	IDWRI	IDWR2	IDEOP	IDATT	IDCBE	IDCBG	IDCBP	LKOPE	LKREA	LKWRI	LKCLO	LKDAT	ECBCT	TIMCT	AXSAV	AXCAL	EBTOI	IDTOE	VCONC
IDERRP	448																											
IDERRI	-																											
IDREAD	330			X			X											X										
IDWRITE	-			X			X												X									
IDWRIT2	-			X			X												X									
IDEOPR	120			X																X								
IDATTN	3F8		X	X	X		X	X												X	X		X					
IDCBENT	380	X												X	X													
IDCBGET	230	X																										
IDCBPUT	-	X																										
LKOPEN	590																											
LKREAD	-																											
LKWRITE	-																											
LKCLOSE	-																											
LKDATTN	-																											
ECBCTRL	78																											
TIMCTRL	130																											
AXSAVE	70																											
AXCALL	-																											
EBTOID	1B8																											
IDTOEB	-																											
VCONC	80																											

The procedures in this package create and manipulate 4 types of control blocks. This section describes the format of these control blocks and how they are used. These control blocks are:

1. IDELCB - Element Control Block: This control block is created by procedure IDOPEN and deleted by IDCLOSE. Only one such control block exists at one time. Each element or set that exists at a given time is represented by an entry in this control block. The control block contains the set membership and auxiliary data for each set or element. Its declaration is:

```
DECLARE 1 IDELCB EXTERNAL CONTROLLED,
            2 #ELEMENTS FIXED BINARY,
            2 #SETS FIXED BINARY,
            2 #BYTES FIXED BINARY,
            2 BUF_TABLE(NENT+1),
              3 ENT_FLAG CHARACTER(4),
              3 ENT_ID FIXED BINARY,
              3 MEMB_PNTRS(NSET),
                4 ENT_INDEX FIXED BINARY,
                4 SET_INDEX FIXED BINARY,
              3 CENT_INDEX FIXED BINARY,
              3 CSET_INDEX FIXED BINARY,
              3 CAUXL CHARACTER(NAUXL);
```

The variables in the dimension fields are the parameters in the calling sequence of IDOPEN.

2. IDATCB - Attention Control Block: This control block is created by procedure IDOPEN and deleted by IDCLOSE. Only one such control block exists at one time. The current status of each interrupt type is contained in this control block. It also contains a pointer to the oldest QATTND control block. Its declaration is:

```
DECLARE 1 IDATCB EXTERNAL CONTROLLED,
            2 #ENTRYS FIXED BINARY,
            2 ATTN_POINTER POINTER,
            2 ATTN_EVENT FIXED BINARY(31,0),
            2 ATTN_TABLE(12),
              3 ATTN_CODES CHARACTER(4),
              3 ATTN_STATE FIXED BINARY,
              3 ATTN_ENTRY FIXED BINARY(31,0);
```

3. IDIOCB - I/O Control Block: This control block is created by procedure IDOPEN and deleted by IDCLOSE. Only one such control block exists at one time. This control block is used to synchronize the transmission of information between the IDIIOM and the SYSTEM/360. Its declaration is:

```
DECLARE 1 IDIOCB EXTERNAL CONTROLLED,
            2 IO_EVENT(2) FIXED BINARY(31,0),
            2 IO_VALUE(2) FIXED BINARY;
```

4. QATTND - Queued Attention Control Block: This control block is created by procedure IDATTN when an attention which is to be queued is received from the IDIIOM. The block is deleted by procedure IDRATTN when the attention is reported. More than one QATTND control block may exist at one time. The declaration of the QATTND control block is:

```
DECLARE 1 QATTND BASED(POINTER_VARIABLE),
          2 STRING CHARACTER(4),
          2 ARRAY(5) FIXED BINARY,
          2 UNTIME FIXED BINARY(31,0),
          2 NEXT POINTER;
```