

PART 1

Graphical Functions

Introduction to Part 1

Contents of Part 1

Part 1 is divided into the following chapters:

Symbolic Constants

Functions for Accessing Graphical Structure

Functions for Reading Attributes

Functions for Writing Attributes

Text Functions

User Interface Functions

Utility Functions

Within each chapter, the functions are listed alphabetically by function name.

Measurement Units

There are 72 points to the inch.

Coordinate System

The center of a page is described by the coordinate pair (0,0). Coordinates above and to the left of center are negative; coordinates below and to the right of center are positive.

Chapter 1

Symbolic Constants

Symbolic Constants

The symbolic constants listed in this chapter are used by many Design/CPN internal functions.

The dictionary of symbolic constants begins on the next page

Color Table Indices

Integer constants which index into the color table.

Values

BLACK
BLUE
BROWN
DK_BROWN
DK_GRAY
DK_GREEN
DK_PURPLE
LT_BLUE
LT_GRAY
LT_GREEN
LT_PURPLE
MED_GRAY
ORANGE
RED
WHITE
YELLOW

Connector Tip Constants

Integer constants to use for determining/setting connector orientations.

Values

BOTHDIR
FROMOTHERNODE
NODIR
TONODE1
TONODE2
TOOTHERNODE

Miscellaneous Constants

Miscellaneous integer constants for use with a variety of functions.

Values

ARROWHEADTYPE
BLOCKSIZE
BOXHEIGHT
BOXWIDTH
CHEADLENGTH
CHEADWIDTH
CONNECTORSHAPE
CONNTEXTBOXHEIGHT
CONNTEXTBOXWIDTH
CORNERRADIUS
ELLHEIGHT
LEFTBRACKETS
LEFTDELIM
NET_STATE
NET_TRANS
ORIENT
POINTSIZ
REF_NOTOWN
REF_OWN
RIGHTBRACKETS
RIGHTDELIM
SEGMENTCURVATURE

Node and Connector Shapes

Symbolic integer constants for node shapes.

Values

CURVESIDECONN
CURVETOPCONN
ELLIPSE
PICTURE
POLYGON
RECTANGLE
RNDRECT
STRAIGHTCONN
WEDGE

Node Types

The different types of nodes as integer constants.

Values

CONNECTOR_TYPE
NODE_TYPE
REGION_TYPE

Object Flags

Integer flags used to read and set properties of objects.

Values

FRONT
INVISIBLE
NOTFRONT
VISIBLE

Object Types

The different types of objects as integer constants.

Values

ANY_TYPE
CONNECTOR_TYPE
DIAGRAM_TYPE
FIELD_TYPE
FORMULA_TYPE
INSCRIP_TYPE
NODE_TYPE
OPER_TYPE
PAGE_TYPE
REGION_TYPE
SOPND_TYPE

Print Options

Constants for use with printing commands.

Description

P_ALL_PAGES
P_PAGE_LIST
P_PAGE_RANGE
P_USE_DIALOG

Text Fonts

Integer constants for identifying various text fonts.

Values

ApplFont
Athens
Cairo
Courier
Geneva
Helvetica
London
LosAngeles
Modern
Monaco
NewYork
Roman
SanFran
Script
Symbol
System
SystemFont
Taliesin
Terminal
Times
TmsRoman
Toronto
Venice

Text Justification

Integer constants for identifying the various text justifications.

Values

Centered
LeftJustification
RightJustification

Text Styles

Integer constants for identifying the various text styles. To get combinations of styles, add together the constituent styles.

Values

Bold
Condense
Extended
Italic
Outline
PlainText
Shadow
Underline

Chapter 2

Functions for Accessing Graphical Structure

Functions for Accessing Graphical Structure

The functions described in this chapter create, modify, and return the attributes of the graphical structure of a net.

The dictionary of functions for accessing graphical structure begins on the next page.

DSStr_AttachPageToNode

Attaches a subpage to a node and its environment.

Synopsis

```
DSStr_AttachPageToNode:  
  {node: int,  
   page: int,  
   matchnodes: int list,  
   repnodes: int list} -> unit  
exception EXStr_AttachPageToNode:unit
```

Description

This routine attaches a node to a page. If the page is a subpage, this routine can be used to create connectors between the 'node' and 'matchnodes' that are designated to represent 'repnodes' (which are the port nodes on the subpage). The arguments 'repnodes' and 'matchnodes' should be empty lists if the page is not a subpage or if no connectors should be created.

Arguments

node	ID of node to contain non-owned refinement.
page	ID of subpage.
matchnodes	List of node IDs to match rep nodes.
repnodes	List of rep node IDs.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_ClosePage

Closes the given page, if it is open.

Synopsis

```
DSStr_ClosePage : int -> unit  
exception EXStr_ClosePage : unit
```

Description

Closes the given page, if it is open. If the page being closed is the current page, it is up to the caller to ensure that there is a valid current page once the close has been done.

Arguments

ID of page to close.

Return value

None.

Exceptions

Raised if the argument is bad.

DSStr_Coarsen

Performs a coarsening of the specified page.

Synopsis

```
DSStr_Coarsen:  
  {page: int,  
   nodes: int list,  
   node: int} -> int  
exception EXStr_Coarsen : unit
```

Description

This routine coarsens a node on the specified page, creating a subpage of this node and moving the nodes identified in 'nodes' to the subpage. If the list is empty, the routine will move all nodes bounded by 'node' to the subpage. If 'node' is 0, the user will be asked to create one. In that case, the application can determine the identity of the coarse node by calling `DSRdAttr_GetParentNode()` with the return value of `DSStr_Coarsen` (the new subpage) as its argument.

Arguments

page	ID of page to coarsen.
nodes	List of node IDs to be moved.
node	ID of node to contain subpage. If 'node' = 0 routine will ask the user to create one.

Return value

Returns ID of subpage.

Exceptions

Raised if unsuccessful.

DSStr_ConnSubGraph

Given a node, calculates the set of nodes in the connected subgraph, that is, those reachable by connector from the given object.

Synopsis

```
DSStr_ConnSubGraph : int -> int list  
exception EXStr_ConnSubGraph : unit
```

Description

Given a node, calculates the set of nodes in the connected subgraph, that is, those reachable by connector from the given object. If the number of nodes is greater than morphmax, an exception is raised.

Arguments

ID of node or region.

Return value

Returns a list of IDs of reachable nodes.

Exceptions

Raised if unsuccessful.

DSStr_CreateConn

Creates a new connector between the specified nodes.

Synopsis

```
DSStr_CreateConn:  
  {page: int,  
   node1: int,  
   node2: int} -> int  
exception EXStr_CreateConn : unit
```

Description

Create a new connector between the specified nodes. The visual attributes of the connector (line pattern and thickness, fill pattern, boundary visibility) are determined by their current default values. In addition, the shape and orientation of the connector are determined by their default values.

Arguments

page	ID of page in which object should be made.
node1	ID of the first node.
node2	ID of the second node.

Return value

Returns ID of connector.

Exceptions

Raised if unsuccessful.

DSStr_CreateLabel

Creates a new label at position (x,y). The label will be sized to fit its text.

Synopsis

```
DSStr_CreateLabel:  
  {page: int,  
   x: int,  
   y: int,  
   w: int,  
   h: int,  
   text: string} -> int  
exception EXStr_CreateLabel : unit
```

Description

This routine creates a new label node.

Arguments

page	ID of page in which label should be made.
x,y	Desired x and y coordinates of label center.
w,h	Initial width and height of label.
text	Text to be placed in label.

Return value

Returns ID of label.

Exceptions

Raised if unsuccessful.

DSStr_CreateLine

Creates a new line between points.

Synopsis

```
DSStr_CreateLine:  
  {page: int,  
   points: int list} -> int  
exception EXStr_CreateLine : unit
```

Description

This routine creates a new line.

Arguments

page	ID of page in which label should be made.
points	x and y coordinates of the origin and end of the line.

Return value

Returns ID of line.

Exceptions

Raised if unsuccessful.

DSStr_CreateNode

Creates a new node.

Synopsis

```
DSStr_CreateNode:  
  {page: int,  
   x: int,  
   y: int,  
   w: int,  
   h: int,  
   shape: int} -> int  
exception EXStr_CreateNode : unit
```

Description

Creates a new node. Node will be placed with center (x,y) specified in model coordinates. Width and height in model coordinates. RECTANGLE or RNDRECT will be a t element (neteltyp=NET_TRANS). ELLIPSE, WEDGE or any POLY will be an s element (neteltyp=NET_STATE). The visual attributes of the node (line pattern and thickness, fill pattern, boundary visibility) are determined by their current default values.

Arguments

page	ID of page for new object.
x	x coordinate of node center.
y	y coordinate of node center.
w	Desired width of node (not applicable to polygons).
h	Desired height of node (not applicable to polygons).
shape	Desired shape of node.

Return value

Returns ID of node.

Exceptions

Raised if unsuccessful.

DSStr_CreatePolygon

Creates a new polygon node.

Synopsis

```
DSStr_CreatePolygon:  
  {page: int,  
   points: int list} -> int  
exception EXStr_CreatePolygon : unit
```

Description

Creates a new polygon node. The visual attributes of the node (line pattern and thickness, fill pattern, boundary visibility) are determined by their current default values.

Arguments

page	ID of page for new object.
points	ID list for the points.

Return value

Returns ID of polygon.

Exceptions

Raised if unsuccessful.

DSStr_DeleteObject

Deletes the designated object from the model structure and frees all space occupied.

Synopsis

```
DSStr_DeleteObject : int -> unit  
exception EXStr_DeleteObject : unit
```

Description

The designated node, region, or connector is deleted from the diagram structure and all space occupied is freed. Any connectors attached to a node or region, and any regions (and their regions and connectors, etc.) of the designated object are also deleted.

Arguments

ID of object to be deleted.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_GetConnOtherEnd

Gets the other end of the specified connector.

Synopsis

```
DSStr_GetConnOtherEnd:  
  {node: int,  
   conn: int} ->  
  {other: int,  
   dir: int}  
exception EXStr_GetConnOtherEnd : unit
```

Description

Gets the other end of the specified connector. Given a node and a connector originating/terminating from/at it, this function will return the node at the other end of the connector. It will also return the direction of the connector relative to the specified node.

Arguments

node ID of the current node.
conn ID of the current connector to 'node'.

Return value

other	ID of the node at the other end of 'conn'.
dir	NODIR If 'conn' has no arrowheads.
TOOTHERNODE	If 'conn' is directed away from 'node'.
FROMOTHERNODE	If 'conn' is directed towards 'node'.
BOTHDIR	If 'conn' has an arrowhead at both ends.

Exceptions

Raised if unsuccessful.

DSStr_GetCurGroup

Gets the members of the current group.

Synopsis

```
DSStr_GetCurGroup : unit -> int list  
exception EXStr_GetCurGroup : unit
```

Description

Gets the members of the current group.

Arguments

None.

Return value

List of node IDs in the current group.

Exceptions

Raised if unsuccessful.

DSStr_GetCurObject

Reads the ID of the currently selected object.

Synopsis

```
DSStr_GetCurObject : unit -> int
```

Description

This routine reads the identification number of the currently selected node, region, or connector. This is the same information that appears in the Get Info box.

Arguments

None.

Return value

Returns the ID of currently selected object.

DSStr_GetCurPage

Reads the ID of the currently selected page.

Synopsis

```
DSStr_GetCurPage : unit -> int
```

Description

This routine reads the identification number of the current page.

Arguments

None.

Return value

Returns the ID of currently selected page.

DSStr_GetDocId

Gets the identification number of the current diagram.

Synopsis

```
DSStr_GetDocId : unit -> int  
exception EXStr_GetDocId : unit
```

Description

The diagram object is the root object of a diagram. This function returns the identification number of the diagram object.

Arguments

None.

Return value

The ID of the diagram object.

Exceptions

Raised if unsuccessful.

DSStr_GetInternalConnList

Gets all the connectors between a set of specified nodes.

Synopsis

```
DSStr_GetInternalConnList : int list -> int list  
exception EXStr_GetInternalConnList : unit
```

Description

Given a set of nodes, this function determines all the connectors which interconnect any two nodes in the specified set. It returns a list of all such connectors.

Arguments

List of node IDs.

Return value

List of connector IDs.

Exceptions

Raised if unsuccessful.

DSStr_GetNodeList

Returns a list of nodes in a specified page.

Synopsis

```
DSStr_GetNodeList : int -> int list  
exception EXStr_GetNodeList : unit
```

Description

Returns a list of nodes in a specified page.

Arguments

ID of page.

Return value

List of node IDs.

Exceptions

Raised if unsuccessful.

DSStr_GetObjectConnList

Given an object (node or region), determines the connectors that are attached to the object.

Synopsis

```
DSStr_GetObjectConnList : int -> int list  
exception EXStr_GetObjectConnList : unit
```

Description

This routine determines the connectors that are attached to the designated node or region.

Arguments

ID of object.

Return value

List of connector IDs.

Exceptions

Raised if unsuccessful.

DSStr_GetObjectInOutLists

Given a node, determines the nodes that are input to it and the nodes that are its outputs.

Synopsis

```
DSStr_GetObjectInOutLists: int ->  
  {ins: int list,  
   outs: int list}  
exception EXStr_GetObjectInOutLists : unit
```

Description

This routine determines which nodes are connected to a specified node and classifies them as inputs or outputs. An input node has either no arrowheads or an arrowhead entering the specified node. An output node has either no arrowheads or an arrowhead pointing away from the specified node.

Arguments

ID of node.

Return value

ins ID list of input connectors.
outs ID list of output connectors.

Exceptions

Raised if unsuccessful.

DSStr_GetObjectRegionList

Determines the IDs of regions in a parent.

Synopsis

```
DSStr_GetObjectRegionList : int -> int list  
exception EXStr_GetObjectRegionList : unit
```

Description

Determines the IDs of regions in a parent.

Arguments

ID of parent.

Return value

ID list of regions.

Exceptions

Raised if unsuccessful.

DSStr_GetPageConnList

Determines the IDs of connectors in a page.

Synopsis

```
DSStr_GetPageConnList : int -> int list  
exception EXStr_GetPageConnList : unit
```

Description

Determines the IDs of connectors in a page.

Arguments

ID of page.

Return value

ID list of connectors.

Exceptions

Raised if unsuccessful.

DSStr_GetPageList

Determines the IDs of pages in a document.

Synopsis

```
DSStr_GetPageList : unit -> int list  
exception EXStr_GetPageList : unit
```

Description

Determines the IDs of pages in a document.

Arguments

None.

Return value

ID list of pages.

Exceptions

Raised if unsuccessful.

DSStr_GetParent

Identifies the parent ID of a page, node, connector or region.

Synopsis

```
DSStr_GetParent : int -> int  
exception EXStr_GetParent : unit
```

Description

Identifies the parent ID of a page, node, connector or region. The parent of a page is always `ROOT_STRUCT`. The grandparent of a node or connector is always a page. The grandparent of a region is a node or a connector, or a region when we allow recursive regions.

Arguments

ID of object whose parent is sought.

Return value

Returns ID of parent.

Exceptions

Raised if unsuccessful.

DSStr_GetTopParent

Finds the node or connector parent of a region.

Synopsis

```
DSStr_GetTopParent : int -> int  
exception EXStr_GetTopParent : unit
```

Description

Starting at the specified region, finds the node or connector parent of that region. Region may be any number of child levels below the node or connector.

Arguments

ID of region.

Return value

Return ID of node/conn.

Exceptions

Raised if unsuccessful; in particular if the argument is not a region.

DSStr_IsPageOpen

Tests whether a page is open or closed.

Synopsis

```
DSStr_IsPageOpen : int -> bool  
exception EXStr_IsPageOpen : unit
```

Description

This routine tests whether a page is open or closed.

Arguments

ID of page to test.

Return value

Returns TRUE if page open, FALSE if closed.

Exceptions

Raised if page is not valid.

DSStr_IsValidObject

Determines whether the object ID represents a valid object in the document.

Synopsis

```
DSStr_IsValidObject : int -> bool
```

Description

This routine ascertains whether an object is a valid object in the diagram.

Arguments

ID of the object.

Return value

Returns TRUE if the system can validate the existence of the object number, FALSE if not.

DSStr_MakeNodeIntoRgn

Changes designated object into a region of designated parent.

Synopsis

```
DSStr_MakeNodeIntoRgn:  
  {obj: int,  
   parent: int} -> unit  
exception EXStr_MakeNodeIntoRgn : unit
```

Description

Changes the designated object into a region of the designated parent. If the object is a node or region, it and all its progeny are transferred to the new parent. (But note: the parent must be in the same page as the object.) If the object or its progeny have connectors attached, the parent must not be a connector or a region of a connector.

Arguments

obj	ID of object to become a region of designated parent.
parent	ID of designated parent for object.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_MakeRgnIntoNode

Calls kernel function to make a region into a node, to unregionize an object.

Synopsis

```
DSStr_MakeRgnIntoNode : int -> unit  
exception EXStr_MakeRgnIntoNode : unit
```

Description

This routine removes a region's link to its parent, making it into a primary node.

Arguments

ID of region to be made into node.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_MoveNodesToPage

Moves a set of nodes to a new page.

Synopsis

```
DSStr_MoveNodesToPage:  
  {nodes: int list,  
   page: int} -> unit  
exception EXStr_MoveNodesToPage : unit
```

Description

Moves a set of nodes to a new page.

Arguments

nodes	ID list of nodes to move. Elements must be nodes.
page	ID of page to move nodes to.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_NewPage

Creates a new page in the document.

Synopsis

```
DSStr_NewPage : int -> int  
exception EXStr_NewPage : unit
```

Description

Creates a new page in the document. The size is set to the current jobwidth and jobdepth, as determined by the last **Page Setup**. The size may subsequently be changed using `DSWtAttr_PageInfo()`.

Arguments

Can be either `RECTANGLE` or `ELLIPSE`.

Return value

Returns ID of page.

Exceptions

Raised if unsuccessful.

DSStr_NewPageWithFlags

Creates a new page in the document.

Synopsis

```
DSStr_NewPageWithFlags:  
  {shape: int,  
   flag: int}      -> int  
exception EXStr_NewPageWithFlags : unit
```

Description

Creates a new page in the document. The size is set to the current jobwidth and jobdepth, as determined by the last Page Setup (see standard file menu). The size may subsequently be changed using DSwtAttr_PageInfo().

Arguments

shape	Can be either RECTANGLE or ELLIPSE.
flag	Can be either VISIBLE, INVISIBLE, FRONT, or NOTFRONT.

Return value

Returns ID of page.

Exceptions

Raised if unsuccessful.

DSStr_PortNodesOnPage

Returns a list of port nodes on a given page.

Synopsis

```
DSStr_PortNodesOnPage : int -> int list  
exception EXStr_PortNodesOnPage : unit
```

Description

Returns a list of port nodes on a given page.

Arguments

ID of page to find ports on.

Return value

ID list of ports.

Exceptions

Raised if unsuccessful.

DSStr_SetCurGroup

Sets the current group, turning group mode on if it is not already on.

Synopsis

```
DSStr_SetCurGroup : int list -> unit  
exception EXStr_SetCurGroup : unit
```

Description

This routine creates the current group, turning group mode 'on' if it is not already on.

Arguments

List of node IDs.

Return value

None.

Exceptions

Raised if list length is greater than GroupMax or if any elements in list are not nodes.

DSStr_SetCurObject

Changes the current object.

Synopsis

```
DSStr_SetCurObject : int -> unit  
exception EXStr_SetCurObject : unit
```

Description

Changes the current object. Turns off aggregate mode if it is currently on.

Arguments

ID of new current object.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_SetCurPage

Sets the current page.

Synopsis

```
DSStr_SetCurPage : int -> unit  
exception EXStr_SetCurPage : unit
```

Description

This routine makes the designated page the current page which MetaDesign operations will now affect. This changes the current window.

Arguments

ID of page.

Return value

None.

Exceptions

Raised if unsuccessful.

DSStr_SetDiagModified

Sets the kernel Modified to TRUE or FALSE.

Synopsis

```
DSStr_SetDiagModified : bool -> unit
```

Description

Sets the kernel Modified to TRUE or FALSE. The flag tells whether to warn users that file needs saving during **Quit**, **Close**, **New**, or **Open** operation.

Arguments

New value for Modified.

Return value

None.

Chapter 3

Functions for Reading Attributes

Functions for Reading Attributes

The functions described in this chapter return the attributes of individual graphical objects within a net, or the global defaults for objects of a particular type.

The dictionary of functions for reading attributes begins on the next page.

DSFile_GetCurrentDiagName

Gets the current diagram name.

Synopsis

```
DSFile_GetCurrentDiagName : unit -> string  
exception EXFile_GetCurrentDiagName : unit
```

Description

This routine gets the current diagram name.

Arguments

None.

Return value

The diagram name.

Exception

Raised if no diagram is currently open.

DSFile_NameDialog

Puts up a file selection dialog to obtain a filename from the user.

Synopsis

```
DSFile_NameDialog:  
  {prompt:string,  
   okbuttonlabel: string,  
   path:string  
   writebox: bool} -> string  
exception EXUI_GetFileName : unit
```

Description

Puts up a file selection dialog (similar to the one put up opening a diagram). The file selection dialog can be used to specify a file name anywhere in the file system.

If the user selects 'OK', the entire pathname of the selected file is returned. If the user selects 'Cancel', an exception is raised.

Arguments

prompt	The prompt string displayed to the user.
okbuttonlabel	Unix only. Appears in the "OK" button.
path	The pathname of the initial directory to be displayed in the dialog.
writebox	Mac only. If TRUE, returns <code>write</code> dialog; if FALSE, returns <code>get</code> dialog.

Return value

Full pathname and filename.

Exception

Raised if user chooses to 'Cancel'.

DSRdAttr_ArrowHeadType

Returns the connector head type

Synopsis

```
DSRdAttr_ArrowHeadType:  
  {conn:int,  
   connend:bool} -> int
```

Description

Returns the connector head type.

Arguments

conn ID of connector.
connend FALSE means node1, TRUE means node2.

Return value

Returns the connector head type for the specified end.

DSRdAttr_ConnOrient

Reads the orientation information for a connector.

Synopsis

```
DSRdAttr_ConnOrient : int -> int  
exception EXRdAttr_ConnOrient : unit
```

Description

Reads the orientation information for a connector. The following predefined symbolic constants provide an interpretation of the return value:

NODIR	unoriented arc
TONODE2	arrowhead at node 2 of connector
TONODE1	arrowhead at node 1 of connector
BOTHDIR	arrows at both ends of connector

Arguments

Object ID of the connector. If 0, return the current default value.

Return value

Returns connector orientation as an integer.

Exception

Raised if argument is invalid.

DSRdAttr_ConnPoints

Reads the points of a connector of type STRAIGHTCONN.

Synopsis

```
DSRdAttr_ConnPoints : int -> int list  
exception RdAttr_ConnPoints : unit
```

Description

Reads the points of a connector of type STRAIGHTCONN.

Arguments

ID of connector.

Return value

List of connector points.

Exception

Raised if not STRAIGHTCONN.

DSRdAttr_ConnProps

Reads attributes for a given connector, or reads the global attributes for connector creation.

Synopsis

```
DSRdAttr_ConnProps: int ->
  {w:int,
   h:int,
   shape:int,
   textw:int,
   texth:int}
exception EXRdAttr_ConnProps : unit
```

Description

Reads attributes for a given connector, or reads the global attributes for connector creation.

Arguments

ID of connector.

Return value

w	Connector head width in points.
h	Connector head height in points.
shape	Straight or curved (=STRAIGHTCONN, CURVETOPCONN, CURVESIDECONN).
textw	Width of text box in points.
texth	Height of text box in points.

Exception

Raised if conn ID is invalid.

DSRdAttr_GetConnEnds

Reads the two node names for a connector.

Synopsis

```
DSRdAttr_GetConnEnds:  
  {conn:int,  
   rgn:bool} ->  
  {node1:int,  
   node2:int}  
exception EXRdAttr_GetConnEnds: unit
```

Description

Reads the two node names for a connector.

Arguments

conn	ID of connector.
rgn	TRUE means return region attachment name; otherwise
return	NODE attachment name.

Note: If connector is not attached to a region, the node and region attachment name will be identical.

Return value

node1	ID of node one.
node2	ID of node two.

Exception

Raised if unsuccessful.

DSRdAttr_GetMaxGroupSize

Returns the maximum allowable size for an aggregate.

Synopsis

```
DSRdAttr_GetMaxGroupSize : unit -> int
```

Description

Returns the maximum allowable size for an aggregate.

Arguments

None.

Return value

The maximum size allowed for a group.

DSRdAttr_GetObjectCenter

Finds the current center coordinates of a page, node, or region.

Synopsis

```
DSRdAttr_GetObjectCenter : int ->  
  {x:int,  
   y:int}  
exception EXRdAttr_GetObjectCenter : unit
```

Description

Finds the current center coordinates of a page, node, or region.

Arguments

ID of object.

Return value

x x coordinate.
y y coordinate.

Exception

Raised if object is not a page, node, or region.

DSRdAttr_GetObjectFlags

Allows the user to read various object flags.

Synopsis

```
DSRdAttr_GetObjectFlags:
    {obj:int,
     flag:int} -> bool
exception EXRdAttr_GetObjectFlags : unit
```

Description

Allows the user to read various object flags. These flags pertain to all model structure objects.

MASK_FLAG	Allows the user to control whether an object is pickable or not. TRUE means not pickable.
OMIT_FLAG	Allows the user to control whether an object and all its substructure should be invisible or not. TRUE means invisible.
NOBOUND_FLAG	Allows the user to control whether an object boundary should be invisible or not. TRUE means invisible.
TEXTCHILD_FLAG	Indicates the presence of hypertext pointers in object text.
TEXTPARENT_FLAG	Indicates the presence of hypertext back pointers.
DT_HAND_FLAG	Indicates the presence of user data.
NOSIZING_FLAG	Indicates an object with text, where the object boundary is determined by the size of the text.

Arguments

obj ID of object.
flag Name of flag.

Return value

TRUE is on.

Exception

Raised if unsuccessful.

DSRdAttr_GetObjectSize

Finds the current width and height of a page, node, or region.

Synopsis

```
DSRdAttr_GetObjectSize: int ->  
  {w:int,  
   h:int}  
exception EXRdAttr_GetObjectSize : unit
```

Description

Finds the current width and height of a page, node, or region.

Arguments

ID of object.

Return value

w Width of object.
h Height of object.

Exception

Raised if object is not a page, node or region.

DSRdAttr_GetObjectSubpage

Gets the subpage ID if the object is a coarse object (node or connector).

Synopsis

```
DSRdAttr_GetObjectSubpage : int -> int  
exception EXRdAttr_GetObjectSubpage : unit
```

Description

Gets the subpage ID if the object is a coarse object (node or connector).

Arguments

ID of node or connector.

Return value

Returns subpage (refinement field) ID if there was a refinement subfield.

Exception

Raised if there was no refinement subfield or if object is other than a node or connector.

DSRdAttr_GetObjectType

Reads the object type information associated with an object's ID.

Synopsis

```
DSRdAttr_GetObjectType : int -> int
```

Description

Reads the object type information associated with an object's ID.
Possible object types are:

```
FIELD_TYPE  
NODE_TYPE  
REGION_TYPE  
CONNECTOR_TYPE
```

Arguments

ID of node.

Return value

Returns the object type as an integer.

DSRdAttr_GetOwnedValue

Reads the node's owned information.

Synopsis

```
DSRdAttr_GetOwnedValue : int -> int  
exception EXRdAttr_GetOwnedValue : unit
```

Description

Reads the node's owned information. If there is representative information, the returned value means:

REF_OWN (0) refinement is owned: coarsenode.
REF_NOTOWN (1) refinement is not owned: attach node.

Otherwise the value is the name of connector on the top page: the port node.

Arguments

ID of node.

Return value

Returns owned value.

Exception

Raised if unsuccessful.

DSRdAttr_GetPageAttr

Gets page (field) attributes.

Synopsis

```
DSRdAttr_GetPageAttr : int ->
  {name: string,
   num: int,
   w: int,
   h: int,
   vis: bool}
exception EXRdAttr_GetPageAttr : unit
```

Description

Gets page (field) attributes.

Arguments

ID of page.

Return value

name	String to put page name in.
num	Page number.
w	Page width (points).
h	Page height (points).
vis	Flag for visible/invisible borders.

Exception

Raised if not a page.

DSRdAttr_GetParentNode

Reads the parent information associated with a page.

Synopsis

```
DSRdAttr_GetParentNode : int -> int  
exception EXRdAttr_GetParentNode : unit
```

Description

Reads the parent information associated with a page.

Arguments

ID of page.

Return value

Returns the ID of parent node of specified page, if page is a sub-page.

Exception

Raised if not a page.

DSRdAttr_GetRegionId

Reads the region identifier associated with a region.

Synopsis

```
DSRdAttr_GetRegionId : int -> int  
exception EXRdAttr_GetRegionId : unit
```

Description

Reads the region identifier associated with a region.

Arguments

ID of object.

Return value

Region ID.

Exception

Raised if not a region.

DSRdAttr_GetRepObject

Reads the representative node or representative connector information for a node or connector, respectively.

Synopsis

```
DSRdAttr_GetRepObject : int -> int  
exception EXRdAttr_GetRepObject : unit
```

Description

Reads the representative node or representative connector information for a node or connector, respectively.

Arguments

ID of node or connector.

Return value

Returns representative value.

Exception

Raised if not a node or connector.

DSRdAttr_GetShape

Reads the object shape information associated with a structure block in the model.

Synopsis

```
DSRdAttr_GetShape : int -> int  
exception EXRdAttr_GetShape : unit
```

Description

Reads the object shape information associated with a structure block in the model. Possible object shapes are :

```
RECTANGLE  
ELLIPSE  
POLYGON  
RNDRECT  
WEDGE  
PICTURE  
REGPOLY  
STRAIGHTCONN  
CURVETOPCONN  
CURVESIDECONN
```

Arguments

ID of node.

Return value

The shape of the node.

Exception

Raised if obj is not a valid ID number.

DSRdAttr_GetTextDefaults

Allows user to read default text attributes.

Synopsis

```
DSRdAttr_GetTextDefaults: unit ->
  {font: int,
   size: int,
   style: int,
   just: int,
   sbar:bool}
```

Description

Allows user to read default text attributes.

Arguments

None.

Return value

font	Font number.
size	Point size.
style	Style.
just	Justification.
sbar	Text scroll bars enabled flag.

DSRdAttr_GetType

Reads the node type information associated with a node.

Synopsis

```
DSRdAttr_GetType : int -> int
```

Description

Reads the node type information associated with a node. Currently, node type may be either NET_STATE (0) or NET_TRANS (1). The user is free to use the high order 35 bits for any purpose. The low order bit is used in Design for automatic grammar checking (states can be connected only to transitions and vice versa).

Arguments

ID of node.

Return value

Returns the work type as an integer.

DSRdAttr_InGroupMode

Reads the global variable for Group Mode.

Synopsis

Description

```
DSRdAttr_InGroupMode : unit -> bool
```

This routine reads the current group state — i.e., whether or not a group is currently enabled.

Arguments

None.

Return value

TRUE if in Group Mode. FALSE if not.

DSRdAttr_NetElementType

Reads the node type information from the structure block or computes it based on the shape of the object, if the object is a region.

Synopsis

```
DSRdAttr_NetElementType : int -> int
```

Description

Reads the node type information from the structure block or computes it based on the shape of the object, if the object is a region.

Arguments

ID of object.

Return value

Returns 32-bit data value.

DSRdAttr_ObjectVisuals

Reads the attributes of an object that affect its appearance.

Synopsis

```
DSRdAttr_ObjectVisuals: int ->
  {line: int,
   thick: int,
   fill: int,
   vis:bool}
exception EXRdAttr_ObjectVisuals : unit
```

Description

Reads the attributes of an object that affect its appearance. If the argument is 0, reads the global attributes that affect future object creation.

Arguments

If 0, reads global attributes. Else, this is the object ID whose attributes are to be read.

Return value

line	Pattern number for border shading.
thick	Border thickness in points (0,1,2,4,6,8,12,16). 0 is laser hairline.
fill	Pattern number for interior fill.
vis	Visibility value: FALSE if border is invisible. TRUE if visible.

For values of 'line' and 'fill', see DSWtAttr_ObjectVisuals().

Exception

Raised if unsuccessful.

DSRdAttr_PageScale

Reads the horizontal and vertical scale of a page.

Synopsis

```
DSRdAttr_PageScale : int ->  
  {xscale: int,  
   yscale: int}  
exception EXRdAttr_PageScale : unit
```

Description

This function reads the horizontal and vertical scale of a page.

Arguments

ID of page. It must be a page.

Return value

xscale	Percent horizontal scale (e.g. 50, 100, 200).
yscale	Percent vertical scale.

Exception

Raised if unsuccessful.

DSRdAttr_PolyDefaults

Allows the user to read the attributes of regular polygons.

Synopsis

```
DSRdAttr_PolyDefaults : unit ->
  {sides: int,
   orient: int}
```

Description

Allows the user to read the attributes of regular polygons.

Arguments

None.

Return value

sides	Number of sides.
orient	Tells whether vertex is above center.

DSRdAttr_PolyPointCount

Reads the number of points in a POLYGON or REGPOLY.

Synopsis

```
DSRdAttr_PolyPointCount : int -> int  
exception EXRdAttr_PolyPointCount : unit
```

Description

This function reads the number of points in a node or region that has shape CONVEX or REGPOLY.

Arguments

ID of node or region.

Return value

Number of points in polygon.

Exception

Raised if shape is not a POLYGON or REGPOLY.

DSRdAttr_PolyPoints

Reads the points of a POLYGON or REGPOLY.

Synopsis

```
DSRdAttr_PolyPoints : int -> int list  
exception EXRdAttr_PolyPoints : unit
```

Description

Reads the points of a POLYGON or REGPOLY.

Arguments

ID of polygon.

Return value

List of coordinate pairs.

Exception

Raised if shape is not a POLYGON or REGPOLY.

DSRdAttr_SegmentCurvature

Returns the segment vertex curvature value for the given connector.

Synopsis

```
DSRdAttr_SegmentCurvature : int -> int
```

Description

Returns the segment vertex curvature value for the given connector. This is a radius in points.

Arguments

ID of connector. If 0, read global curvature value.

Return value

This value controls the curvature between segments in a segmented connector. If the high order bit is set, then the curvature is the value in the lower 31 bits. The low 31 bits are set to 0 for maximum curvature. If the high order bit is not set, then no curve is drawn between segments.

DSRdAttr_SelectableFlag

Reads the current value of the Global Selectable flag.

Synopsis

```
DSRdAttr_SelectableFlag : unit -> bool
```

Description

Reads the current value of the Global Selectable flag. This flag affects future creation of objects. If it is set, newly created objects will not be pickable.

Arguments

None.

Return value

TRUE if flag is set, FALSE if not set.

DSRdAttr_TextPointSize

This function reads the text point size of an object.

Synopsis

```
DSRdAttr_TextPointSize : int -> int  
exception EXRdAttr_TextPointSize : unit
```

Description

This function reads the text point size of an object.

Arguments

ID of object whose point size to read.

Return value

Returns the point size if object has text.

Exception

Raised if object has no text.

Chapter 4

Functions for Writing Attributes

Functions for Writing Attributes

The functions described in this chapter modify the attributes of individual graphical objects within a net, or the global defaults for objects of a particular type.

The dictionary of functions for writing attributes begins on the next page.

DSWtAttr_AdjustObjectSize

Allows the width and height of a page, node or region to be changed.

Synopsis

```
DSWtAttr_AdjustObjectSize :  
  {obj: int,  
   w:int,  
   h:int} -> unit  
exception EXWtAttr_AdjustObjectSize : unit
```

Description

This routine sets the width and height of a page, node, or region.

Arguments

obj	ID of object.
w	New width.
h	New height.

Return value

None.

Exception

Raised if object is not a PAGE, NODE or REGION.

DSWtAttr_ConnCurvature

Writes the curvature value for the given connector.

Synopsis

```
DSWtAttr_ConnCurvature :  
  {conn: int,  
   curve: int} -> unit
```

Description

Writes the curvature value for the given connector. This value controls the curvature between segments in a segmented connector. If the high order bit is set, then the curvature is the value in the lower 31 bits. The low 31 bits are set to 0 for maximum curvature. If the high order bit is not set, then no curve is drawn between segments.

Arguments

conn	ID of connector. If 0, write the default curvature value.
curve	Curvature value in points.

Return value

None.

DSWtAttr_ConnEndIds

Changes the connector ends information for a connector.

Synopsis

```
DSWtAttr_ConnEndIds :  
  {conn: int,  
   node1: int,  
   node1rgn: int,  
   node2: int,  
   node2rgn: int} -> unit  
exception EXWtAttr_ConnEndIds : unit
```

Description

This routine changes the node or region ends of a connector.

Arguments

conn	ID of connector.
node1	ID of node one attachment.
node1rgn	ID of node one secondary attachment. (Must be equal to node1 or a region of node1.)
node2	ID of node two attachment.
node2rgn	ID of node two secondary attachment. (Must be equal to node2 or a region of node2.)

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_ConnOrient

Changes the orientation information for a connector, or the default connector orientation.

Synopsis

```
DSWtAttr_ConnOrient :  
  {conn: int,  
   orient: int} -> unit  
exception EXWtAttr_ConnOrient : unit
```

Description

This routine changes the orientation information of a connector, or the global orientation for connector creation.

Arguments

conn	ID of connector. If 0, change the default orientation.
orient	New orientation. Orientation values are as follows:
NODIR (0)	unoriented arc
TONODE2 (1)	arrowhead at node 2 of connector
TONODE1 (2)	arrowhead at node 1 of connector
BOTHDIR (3)	arrows at both ends of connector

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_ConnVisuals

Writes attributes for a given connector, or writes the global attributes for connector creation.

Synopsis

```
DSWtAttr_ConnVisuals :  
  {conn: int,  
   w: int,  
   h: int,  
   shape: int,  
   textw: int,  
   texth: int} -> unit  
exception EXWtAttr_ConnVisuals : unit
```

Description

This routine writes attributes for a given connector, or writes the global attributes to be used for connector creation.

Arguments

conn	ID of connector. If 0, then write the global attributes.
w	Connector head width in points.
h	Connector head height in points.
shape	Straight or curved (STRAIGHTCONN, CURVETOPCONN, CURVESIDECONN).
textw	Width of text box in points.
texth	Height of text box in points.

Return value

None.

Exception

Raised if connector ID is not valid.

DSWtAttr_LineThickness

Allows the user to control thickness of lines.

Synopsis

```
DSWtAttr_LineThickness :  
  {obj: int,  
   thick: int} -> unit  
exception EXWtAttr_LineThickness : unit
```

Description

This routine sets the thickness of an object boundary.

Arguments

obj	ID of object.
thick	Thickness in model units.

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_LineType

Allows the user to control line type of an object boundary.

Synopsis

```
DSWtAttr_LineType:  
  {obj: int,  
   line: int} -> unit  
exception EXWtAttr_LineType : unit
```

Description

This routine sets the pattern used to paint an object's boundary.

Arguments

obj ID of object.
line Integer code for line type:

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_ObjectFillType

Allows the user to control fill type of an object or arrowhead.

Synopsis

```
DSWtAttr_ObjectFillType :  
  {obj: int,  
   fill: int} -> unit  
exception EXWtAttr_ObjectFillType : unit
```

Description

This routine sets the pattern used to fill an object.

Arguments

obj ID of object.
fill Integer code for fill type. (See DSWtAttr_Object Visuals().)

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_ObjectFlags

Allows the user to write various object flags.

Synopsis

```
DSWtAttr_ObjectFlags :  
  {obj: int,  
   flag: int,  
   value: bool} -> unit  
exception EXWtAttr_ObjectFlags : unit
```

Description

Allows the user to write various object flags. These flags pertain to all objects.

MASK_FLAG	Allows the user to control whether an object is pickable or not. TRUE means not pickable.
OMIT_FLAG	Allows the user to control whether an object and all its substructure should be invisible or not. TRUE means invisible.
NOBOUND_FLAG	Allows the user to control whether an object boundary should be invisible or not. TRUE means invisible.
TEXTCHILD_FLAG	Indicates the presence of pointers in object text.
TEXTPARENT_FLAG	Indicates the presence of back pointers.
DT_HAND_FLAG	Indicates the presence of user data.
NOSIZING_FLAG	Indicates an object with text, where the object boundary is determined by the size of the text.

Arguments

obj	ID of object.
flag	Name of flag.
value	TRUE is on.

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_ObjectPosition

Moves the designated node or region to coordinate position (x,y).

Synopsis

```
DSWtAttr_ObjectPosition:  
  {obj: int,  
   x: int,  
   y: int}      -> unit  
exception EXWtAttr_ObjectPosition : unit
```

Description

This routine moves the designated node or region to the coordinate position (x, y) which is specified by the argument 'x' and 'y', given in world units.

Arguments

obj	ID of object to be moved.
x	x coordinate of new center position.
y	y coordinate of new center position.

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_ObjectVisuals

Sets the attributes of an object that affect its appearance.

Synopsis

```
DSWtAttr_ObjectVisuals :
  {obj: int,
   line: int,
   thick: int,
   fill: int,
   vis: bool} -> unit
exception EXWtAttr_ObjectVisuals : unit
```

Description

Sets the attributes of an object that affect its appearance. If the 'obj' parameter is 0, set the global attributes that affect future object creation.

Arguments

obj If 0, set global attributes. Else, this is the ID of object whose attributes to set.

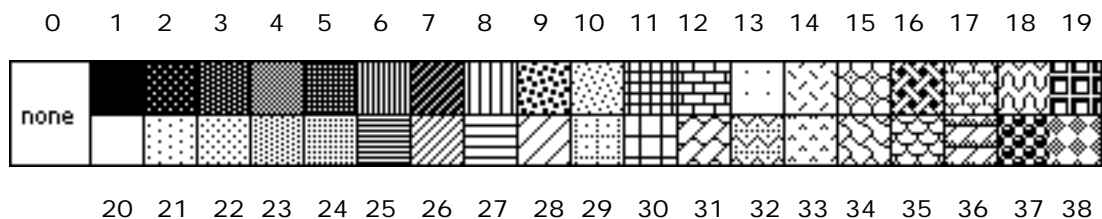
line Pattern number for border shading. For Macintosh versions, specify a pattern number for the border shading, using the values shown below for 'fill'.

Line types in the Sun version are:

- 0 solid
- 1 dashed
- 2 long-dashed
- 3 dotted
- 4 dotted/dashed

thick Border thickness in points (0 ,1 ,2, 4 ,6, 8, 12, 16).

fill Pattern number for interior fill:



vis FALSE if border to be invisible; TRUE if visible.

Graphical Functions

Return value

None.

Exception

Raised if object ID is bogus.

DSWtAttr_PageInfo

Changes page attributes.

Synopsis

```
DSWtAttr_PageInfo :  
  {page: int,  
   name: string,  
   num: int,  
   w: int,  
   h: int,  
   vis: bool} -> unit  
exception EXWtAttr_PageInfo : unit
```

Description

This routine changes page attributes.

Arguments

page	ID of page.
name	String name of page.
num	Page number.
w	Page width (points).
h	Page height (points).
vis	Flag for visible/invisible borders.

Return value

None.

Exception

Raised if not a field or not a structure.

DSWtAttr_RegionId

Writes a user-designated region type for a region.

```
Synopsis
DSWtAttr_RegionId :
  {obj: int,
   rgnid: int} -> unit
exception EXWtAttr_RegionId : unit
```

Description

This routine can be used to write a user-designated type for a region. This is useful if you wish to distinguish between different kinds of regions. You may use such a type to mark the region for any purpose.

Arguments

obj	ID of object.
rgnid	New region identifier.

Return value

None.

Exception

Raised if unsuccessful.

Related Function

DSRdAttr_GetRegionId

DSWtAttr_RegularPolyInfo

Allows the user to write the attributes of regular polygons.

Synopsis

```
DSWtAttr_RegularPolyInfo :  
  {sides: int,  
   vertexup: bool} -> unit  
exception EXWtAttr_RegularPolyInfo : unit
```

Description

This routine sets the attributes used for regular polygon creation. If the orientation flag is TRUE, the vertex of a new regular polygon is placed directly above the center of the polygon; if FALSE, a side is placed directly above the center.

Arguments

sides Number of sides.
vertexup Orientation flag. Tells whether vertex is above center.

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_RepNodeId

Writes the represented node or represented connector information for a node or connector, respectively.

Synopsis

```
DSWtAttr_RepNodeId :  
  {obj: int,  
   repval: int} -> unit  
exception EXWtAttr_RepNodeId : unit
```

Description

Writes the child page (refine) ID of a node or a connector. This turns the node into the parent of a subpage, i.e., into a coarse node. Remember the subpage's parent must be the node (see `DSWtAttr_SetPageParent()`). A connector is made into a coarse connector to a subpage if 'repval' is nonzero. It is unmade as a coarse connector if 'repval' is zero.

Arguments

obj	ID of node or connector whose child the page will be.
repval	Child page ID value to write.

Return value

None.

Exception

Raised if unsuccessful.

DSWtAttr_SetConnPoints

Writes the points of a connector of type STRAIGHTCONN, replacing the old set of points.

Synopsis

```
DSWtAttr_SetConnPoints :  
    {conn: int,  
     points: int list} -> unit  
exception EXWtAttr_SetConnPoints : unit
```

Description

This routine writes the points of a connector of type CONN1, replacing the old set of points. User must call DSUI_Redraw() or DSUI_UpdateCurrentPage() to see the changes.

Arguments

conn	ID of connector.
points	List of coordinate pairs (points). Can't exceed MAXPOINTS*2.

Return value

None.

Exception

Raised if not STRAIGHTCONN.

DSWtAttr_SetDefaultSelectable

Writes the current value of the global Selectable flag.

Synopsis

```
DSWtAttr_SetDefaultSelectable : bool -> unit
```

Description

Writes the current value of the global Selectable flag. This flag affects future creation of objects. If it is set, newly created objects will not be pickable.

Arguments

TRUE if flag should be set.
FALSE if it should not.

Return value

None.

DSWtAttr_SetPetriNodeType

Writes the Petri node type information associated with a node.

Synopsis

```
DSWtAttr_SetPetriNodeType :  
  {obj: int,  
   objtype: int} -> unit
```

Description

Writes the Petri node type information associated with a node. Currently, node type may be either NET_STATE (0) or NET_TRANS (1). The user is free to use the high order 31 bits for any purpose. The low order bit is used in Design for automatic grammar checking (states can be connected only to transitions and vice versa).

Arguments

obj	ID of object.
objtype	New node type.

Return value

None.

DSWtAttr_SetPolyPoints

Writes the points of a POLYGON or REGPOLY, replacing the old set of points.

Synopsis

```
DSWtAttr_SetPolyPoints :  
  {poly: int,  
   points: int list} -> unit  
exception EXWtAttr_SetPolyPoints : unit
```

Description

This routine writes the points of a POLYGON or REGPOLY, replacing the old set of points.

Arguments

poly	ID of polygon.
points	List of coordinate pairs.

Return value

None.

Exception

Raised if the object is not POLYGON or REGPOLY.

Chapter 5

Text Functions

Text Functions

The functions described in this chapter read and write text, set text attributes, and perform miscellaneous operations relating to text.

The dictionary of text functions begins on the next page.

DSText_Append

Adds text to an object. If the object already contains text, the new text is appended to it.

Synopsis

```
DSText_Append :  
  {obj: int,  
   text: string} -> unit  
exception EXText_Append : unit
```

Description

Adds text to an object. If the object already contains text, the new text is appended to it. At most 4000 characters can be added to an object at each call. Any additional text will be truncated.

Arguments

obj	ID of object to which to append text.
text	Text to be appended to existing text.

Return value

None.

Exception

Raised if unsuccessful.

DSText_Get

Reads text from the Text Edit record associated with an object ID.

Synopsis

```
DSText_Get : int -> string  
exception EXText_Get : unit
```

Description

Reads text from the Text Edit record associated with an object ID. Text may be associated with nodes, connectors, or regions. At most 4000 characters will be returned: additional text in the object will be ignored.

Arguments

obj ID of object from which to read text.

Return value

Returns the text.

Exception

Raised if unsuccessful.

DSText_GetLength

Determines length of text associated with an object ID.

Synopsis

```
DSText_GetLength : int -> int  
exception EXText_GetLength : unit
```

Description

Determines length of text associated with an object ID. Text may be associated with nodes, connectors, or regions.

Arguments

ID of object from which text length is to be read.

Return value

Length of text.

Exception

Raised if unsuccessful.

DSText_GetTextParent

Returns the text parent of a node, connector, or region.

Synopsis

```
DSText_GetTextParent : int -> int  
exception EXText_GetTextParent : unit
```

Description

Returns the text parent of a node, connector, or region.

Arguments

ID of node, connector, or region.

Return value

Returns the text parent of a valid node, connector, or region.

Exception

Raised if the object is not a node, connector, or region or if there is no text parent.

DSText_IsModeOn

Reads the current text state.

Synopsis

```
DSText_IsModeOn : unit -> bool
```

Description

This routine reads the current text state. When text is on, the end user may edit, select and search for text in the current object.

Arguments

None.

Return value

Returns TRUE if text on, FALSE if text off.

DSText_MaxLineLength

Returns the width of the longest line of text of a text record.

Synopsis

```
DSText_MaxLineLength : int -> int
```

Description

Returns the width of the longest line of text of a text record.

Arguments

ID of object with text record.

Return value

Returns longest length.

DSText_Put

Writes the supplied text into the Text record associated with a model ID. Any text currently in the model ID is deleted.

Synopsis

```
DSText_Put :  
  {obj: int,  
   text: string} -> unit  
exception EXText_Put : unit
```

Description

Writes the supplied text into the Text record associated with a model ID. Any text currently in the model ID is deleted. Text may be associated with nodes, connectors, or regions. At most 4000 characters can be written: additional characters will be truncated.

Arguments

obj	ID of object in which to store text.
text	Text supplied.

Return value

None.

Exception

Raised if unsuccessful.

DSText_SetAttr

Allows user to write text attributes for a given object.

Synopsis

```
DSText_SetAttr:  
  {obj: int,  
   font: int,  
   size: int,  
   style: int,  
   just: int} -> unit
```

Description

This routine writes the text attributes of a node, connector, or region. It allows you to set the font, point size, style, and justification. See Text Fonts, Text Justification, and Text Styles in Chapter 1, “Symbolic Constants.”

Font numbers are both platform- and printer-dependent.

Arguments

obj	ID of object.
font	Font number.
size	Point size.
style	Style.
just	Justification.

Return value

None.

DSText_SetDefaultFont

Changes the default font.

Synopsis

```
DSText_SetDefaultFont: int -> unit
```

Description

This routine changes the default font. See Text Fonts in Chapter 1, “Symbolic Constants.”

Arguments

This is best given as one of the constants above.

Return value

None.

DSText_SetDefaultJust

Changes the default text justification.

Synopsis

```
DSText_SetDefaultJust : int -> unit
```

Description

This routine changes the default text justification. See Text Justification in Chapter 1, “Symbolic Constants.”

Arguments

This is best given as one of the constants above.

Return value

None.

DSText_SetDefaultSize

Changes the default point size.

Synopsis

```
DSText_SetDefaultSize : int -> unit
```

Description

This routine changes the default point size. The common point sizes are 9, 12, 18, and 24.

Arguments

The size of default point.

Return value

None.

DSText_SetDefaultStyle

Changes the default text style.

Synopsis

```
DSText_SetDefaultStyle : int -> unit
```

Description

This routine changes the default text style. See Text Styles in Chapter 1, “Symbolic Constants.”

Arguments

This is best given as one of the constants above. For combinations of these styles, the user has only to add them together. E.g., to get both bold and italics: `style = Bold + Italic`.

Return value

None.

DSText_SetMode

Writes the current text state.

Synopsis

```
DSText_SetMode : bool -> unit
```

Description

This routine turns text on or off. Turning text on allows the user to edit, select, and search for text in the current object.

Arguments

New state.

Return value

None.

Chapter 6

User Interface Functions

User Interface Functions

The functions described in this chapter allow the system to query the user, and the user to direct the system.

The dictionary of user interface functions begins on the next page.

DSUI_Align

Aligns an object with respect to other objects.

Synopsis

```
DSUI_Align:  
  {obj:int,  
   aligntype:int,  
   ref1:int,  
   ref2:int} -> unit
```

Description

Aligns 'obj' with respect to 'ref1' and 'ref2'. Use 'aligntype' to specify how 'obj' is to be aligned.

Arguments

obj ID of object to be aligned.
aligntype Specifies what kind of alignment to perform:

ALN_H	ALN_TB
ALN_V	ALN_BT
ALN_LL	ALN_BB
ALN_LR	ALN_CENT
ALN_RL	ALN_BETW
ALN_RR	ALN_PROJ
ALN_TT	ALN_RADIAL

ref1 ID of first reference object. If this is 1, the user will be asked to pick an object.

ref2 ID of second reference object; required only for BETWEEN and PROJECTION. If this is required and is 1, the user will be asked to pick an object. Set this to 0 for all other alignment types.

Return value

None.

DSUI_AskUserToSelectPage

Prompts user to select a page.

Synopsis

```
DSUI_AskUserToSelectPage : unit -> int  
exception EXUI_AskUserToSelectPage : unit
```

Description

This routine draws the tree structure of pages on the screen and lets the user specify a particular page by mousing down on the page name. The selected page's identification is returned; use the function `DSStr_SetCurPage()` to make this page the current page.

Arguments

None.

Return value

The selected page's ID.

Exception

Raised if no page selected.

DSUI_AutoPan

Pans the current page to make the specified object visible.

Synopsis

```
DSUI_AutoPan :  
  {obj:int,  
   center:bool} -> unit
```

Description

This routine scrolls the current page to make the specified node, region, or connector visible. The object must be on the current page to be made visible.

Arguments

object	ID of object.
center	True means pan so that the object is centered. False means just make sure object is visible.

Return value

None.

DSUI_BeepUser

Makes the machine BEEP.

Synopsis

```
DSUI_BeepUser : int -> unit
```

Description

This routine causes the system to beep for a duration of time specified by the argument.

Arguments

Duration of beep in 1/60 sec units.

Return value

None.

DSUI_ChangeCursor

Changes the appearance of the cursor.

Synopsis

```
DSUI_ChangeCursor : int -> int
```

Description

This function changes the appearance of the cursor to that specified in the call; cursors are identified by integer codes. The previous cursor ID is the result.

Arguments

ID of new cursor.

Return value

ID of old cursor.

DSUI_CheckBounds

Checks the user's desired value of a dialog item against the minimum and maximum, and requires the user to adjust it if necessary.

Synopsis

```
DSUI_CheckBounds :  
  {value: int,  
   min: int,  
   max: int,  
   strnum: int} -> bool
```

Description

This function checks the user's desired value of a dialog item against the minimum and maximum, and requires the user to adjust it if necessary. The signs of the numbers are taken into account.

Arguments

value	Value to test.
min	Minimum allowed value.
max	Maximum allowed value.
strnum	Resource string ID representing name of dialog item being checked.

Return value

TRUE if 'value' is between 'min' and 'max'.
FALSE otherwise

DSUI_Cleanup

Cleans up the graphical structure of specified page.

Synopsis

```
DSUI_Cleanup : int -> bool
```

Description

This function redraws a page, recalculating the screen values of all objects from the values stored in world units.

Arguments

ID of page to be cleaned up.

Return value

TRUE is on.

DSUI_Duplicate

Duplicates a given set of nodes on a given page, preserving their positions.

Synopsis

```
DSUI_Duplicate :  
  {page:int,  
   nodes:int list} -> int list  
exception EXUI_Duplicate : unit
```

Description

This routine duplicates a given set of nodes on a given page, preserving their positions. Regions of nodes in the list are also duplicated, as well as any connectors that connect two nodes in the list to each other.

Arguments

page	ID of page where nodes reside.
nodes	ID list of nodes to be duplicated.

Return value

ID list of new nodes obtained by duplication.

Exception

Raised if unsuccessful.

DSUI_GetIntegerValue

To prompt the user for an integer value.

Synopsis

```
DSUI_GetIntegerValue :  
  {prompt:string,  
   def:int} -> int  
exception EXUI_GetIntegerValue : unit
```

Description

A dialog box will appear with the prompt message (uneditable) and the default answer (editable). The user may edit the default. The dialog is terminated by picking 'accept' or 'cancel'. (Carriage return is treated like accept.)

Arguments

prompt	Prompt string.
def	The default value.

Return value

The integer value typed by the user.

Exception

Raised if user chooses to cancel or there is an error.

DSUI_GetString

To prompt the user for a string.

Synopsis

```
DSUI_GetString :  
    {prompt:string,  
     def:string} -> string  
exception EXUI_GetString : unit
```

Description

A dialog box will appear with the prompt message (uneditable) and the default answer (editable). The user may edit the default. The dialog is terminated by picking 'accept' or 'cancel'. (Carriage return is treated like accept.)

Arguments

prompt	Prompt string.
def	Default string.

Return value

The string typed by the user.

Exception

Raised if user chooses to cancel or there is an error.

DSUI_GetUserYesOrNo

To prompt the user to make a two-way decision.

Synopsis

```
DSUI_GetUserYesOrNo : string -> bool
```

Description

A dialog box will appear with the prompt message (uneditable) and yes or no as the possible responses. The user may pick yes or no or type carriage return for yes, or type the key 'y' for yes, 'n' for no.

Arguments

The prompt string.

Return value

Returns TRUE if user selects YES, FALSE if user selects NO.

DSUI_Indicate

Indicates a group of nodes and/or regions on the screen.

Synopsis

```
DSUI_Indicate :  
  {nodes:int list,  
   on:bool} -> unit  
exception EXUI_Indicate : unit
```

Description

Indicates a group of nodes and/or regions in the same way that the current group is shown on the screen. (Boundaries are drawn around all members, using XOR logic.) Remember to turn off the boundaries, by calling this routine with `on = FALSE`, before you do something to change the screen.

Arguments

<code>nodes</code>	ID list of nodes.
<code>on</code>	TRUE means draw the group style boundaries.

Return value

None.

Exception

Raised if any object is not a node or region, 'nodes' is empty, or exceeds maximum allowed.

DSUI_IndicateObject

Draws the dot handles for the specified object.

Synopsis

```
DSUI_IndicateObject :  
  {obj:int,  
   on:bool} -> unit
```

Description

This function indicates the specified object by drawing dot handles around its boundary.

Arguments

object	ID of the object to indicate.
on	Tells whether to turn the handles on or off.

Return value

None.

DSUI_MakePageVisible

Makes a page visible if is not currently visible.

Synopsis

```
DSUI_MakePageVisible:  
{page:int,  
 front:bool} -> unit
```

Description

This routine makes a page visible if is not currently visible. The page may be made to be the front page or not.

Arguments

page	ID of page to make visible.
front	TRUE if page to be placed in front, FALSE if not.

Return value

None.

DSUI_Merge

Merges a group of nodes into a target node.

Synopsis

```
DSUI_Merge :  
  {nodes:int list,  
   node:int} -> unit  
exception EXUI_Merge : unit
```

Description

This function takes the given list of nodes and merges them into the target node. The nodes to be merged and the target node must be on the same page.

Arguments

nodes	ID list of nodes to be merged.
node	ID of target node to merge into.

Return value

None.

Exception

Raised if unsuccessful.

DSUI_NoUndo

To prevent **Undo** operations.

Synopsis

```
DSUI_NoUndo : unit -> unit
```

Description

This function calls the Kernel to cancel **Undo**. The MetaDesign commands **Cut**, **Clear**, **Copy**, **Paste**, and **Align** can be undone via the **Undo** command in the Edit menu. `DSUI_NoUndo()` removes the user's access to the last **Undo** action.

Arguments

None.

Return value

None.

DSUI_PreventObjectAdjust

To prevent user from performing size adjustments on all objects.

Synopsis

```
DSUI_PreventObjectAdjust : bool -> unit
```

Description

Normally, a user is allowed to adjust the size of objects or the shape of polygons or connectors, by mousing down on the dot handles used to indicate the currently selected object. This facility can be enabled/disabled by this function.

If you wish to make an individual object non-adjustable, set its `NOSIZING_FLAG` using `DSWtAttr_ObjectFlags()`.

Arguments

TRUE means prevent adjustments.
FALSE means allow adjustments.

Return value

None.

DSUI_Redraw

Redraws the specified object.

Synopsis

```
DSUI_Redraw : int -> unit
```

Description

This routine redraws a specified node, connector, region, or page and all of its regions. It does nothing if object does not identify one of these types of objects. No redrawing occurs if object is not visible. This routine does not change the current window.

Arguments

The ID of the object to redraw.

Return value

None.

DSUI_RestoreStatusBar

Clears any message in the status bar, restoring the normal display.

Synopsis

```
DSUI_RestoreStatusBar : unit -> unit
```

Description

This routine clears a message in the status bar, returning the display to normal. You should always call this at some point after displaying a message with `DSUI_SetStatusBarMessage()`.

Arguments

None.

Return value

None.

DSUI_SelectObject

Prompts the user to select an object.

Synopsis

```
DSUI_SelectObject :  
  {objtype:int,  
   override:bool} -> int  
exception EXUI_SelectObject : unit
```

Description

Uses the Design select facility to allow the user to select an object. When the user moves the cursor over any object, it will flash. When the user clicks the mouse, the flashing object will be selected. The user may hide flashing objects which may obscure the one desired by pressing the space bar.

Arguments

- objtype The type of the object to be picked. Types can be summed together, i.e.: NODE_TYPE + REGION_TYPE, and either will be pickable. See Chapter 1, "Symbolic Constants," for object types.
- override TRUE if unpickable flags in objects should be disregarded for this picking, FALSE if unpickables should not be chooseable.

Return value

Returns the ID of the object selected.

Exception

Raised if unsuccessful.

Related Functions

```
DSUI_SetStatusBarMessage()  
DSWtAttr_ObjectFlags()  
DSRdAttr_GetObjectFlags()
```

DSUI_SetObjectIndication

To enable/disable the indicate dot feature for all objects.

Synopsis

```
DSUI_SetObjectIndication : bool -> unit
```

Description

This routine disables or enables the indication, with dot handles, of the current object.

Arguments

TRUE implies enable.
FALSE implies disable.

Return value

None.

DSUI_SetRepConnDeleteMode

Sets the treatment of represented connector deletion.

Synopsis

```
DSUI_SetRepConnDeleteMode : bool -> unit
```

Description

This function sets an internal flag governing the deletion of coarse connectors. This controls whether coarse connectors are deleted on the parent page when the corresponding port node is deleted on the subpage or when the subpage itself is deleted. The default is to delete the coarse connector.

Arguments

TRUE implies coarse connectors are deleted when their port nodes are deleted.

FALSE implies they are not deleted.

Return value

None.

DSUI_SetStatusBarMessage

Puts a message in the status bar.

Synopsis

```
DSUI_SetStatusBarMessage : string -> unit
```

Description

This routine puts a message in the status bar. The user should issue a `DSUI_RestoreStatusBar()` call when finished with the message.

Arguments

The message to display in the status bar.

Return value

None.

Related Function

`DSUI_RestoreStatusBar()`

DSUI_Spread

Spreads a grouping of three or more nodes to achieve equal space between them.

Synopsis

```
DSUI_Spread:  
  {nodes:int list,  
   v:bool,  
   h:bool} -> unit  
exception EXUI_Spread : unit
```

Description

Spreads a grouping of three or more nodes to achieve equal space between them.

Arguments

nodes	ID list of nodes.
v	TRUE to spread vertically.
h	TRUE to spread horizontally.

Return value

None.

Exception

Raised if there are less than three nodes in the list or if the list contains types other than nodes.

DSUI_UpdateCurrentPage

Redraws the current page on the screen.

Synopsis

```
DSUI_UpdateCurrentPage : unit -> unit
```

Description

Redraws the current page on the screen, showing all changes made since leaving the main event loop. Use this routine when you need to show the user exactly what the current page looks like, without returning to the main event loop.

Arguments

None.

Return value

None.

DSUI_UserAckMessage

Displays user supplied message as a modal dialog.

Synopsis

```
DSUI_UserAckMessage: string -> unit
```

Description

Displays user supplied message (not necessarily an error) as a modal dialog. Waits for user to acknowledge the message before returning.

Arguments

The message to be displayed to the user.

Return value

None.

Chapter 7

Utility Functions

Utility Functions

The functions described in this chapter perform a variety of miscellaneous services.

The dictionary of utility functions begins on the next page.

DSUtil_DrawArc

Calculates and draws a circular curve consisting of line segments from a starting point to an ending point around a given center point.

Synopsis

```
DSUtil_DrawArc :  
  {startpoint: int * int,  
   endpoint: int * int,  
   centerpoint: int * int,  
   rev: bool} -> unit
```

Description

Points must be in model coordinates. Designed to aid in the formation of curves and circles in connector head formation.

Arguments

startpoint	Starting point.
endpoint	Ending point.
centerpoint	Circle midpoint.
rev	If TRUE, draw curve with opposite orientation.

Return value

None.

DSUtil_GetConnClipPoint

Given a connector and an object at one end (secondary attachment object), gets the clip point and puts it directly into the points vector.

Synopsis

```
DSUtil_GetConnClipPoint :  
  {conn: int,  
   obj: int} ->  
  {x: int,  
   y: int}  
exception EXUtil_GetConnClipPoint : unit
```

Description

Given a connector and an object at one end (secondary attachment object), gets the clip point and puts it directly into (x,y).

Arguments

conn ID number of connector.
obj ID number of object at end of connector whose clip point is to be determined. Must be secondary attachment.

Return value

Points.

Exception

Raised if unsuccessful.

DSUtil_IsALabel

Tells whether an object is a label.

Synopsis

```
DSUtil_IsALabel : int -> bool
```

Description

This function determines if an object is a label.

Arguments

ID number of object.

Return value

TRUE if object is a label, FALSE otherwise.

DSUtil_LineToInCoords

Draws a line from the current pen position to a given point.

Synopsis

```
DSUtil_LineToInCoords : int * int -> unit
```

Description

Draws a line from the current pen position, using the current linestyle, to a given point. Designed to facilitate drawing of user-defined arrowheads.

Arguments

Point to draw line to. Must be in model coordinates, referenced to window center (origin).

Return value

None.

DSUtil_Pause

Pauses the specified number of time units. One unit equals 1/60 of a second.

Synopsis

```
DSUtil_Pause : int -> unit
```

Description

This routine pauses the specified number of time units. One unit equals 1/60 of a second.

Arguments

How many 1/60 second units to pause.

Return value

None.

DSUtil_PointInObject

Determines if the given point is inside one of the given object types on the current page.

Synopsis

```
DSUtil_PointInObject :  
  {x: int,  
   y: int,  
   types: int} -> int  
exception EXUtil_PointInObject : unit
```

Description

This routine determines if a given point is inside an object of the specified type on the current page.

Arguments

x,y	World coordinates of point in question.
types	Any combination of NODE_TYPE, REGION_TYPE, CONNECTOR_TYPE.

Return value

Returns ID of object if the point is inside an object.

Exception

Raised if the point not inside an object.

DSUtil_PointsToWorld

Converts from points (72 to the inch) to model units.

Synopsis

```
DSUtil_PointsToWorld : int -> int
```

Description

This routine converts from points (72 to the inch) to world units.

Arguments

Value in points.

Return value

Returns equivalent in model units.

DSUtil_PrintPages

Allows the OA application to print any set of pages with or without using the print dialog.

Synopsis

```
DSUtil_PrintPages:  
  {opt: int,  
   pages: int list,  
   nums: int list} -> unit  
exception EXUtil_PrintPages : unit
```

Description

One of four types of print operations is specified. Each operation has its own requirements for more information about the print. This function allows the information to be passed as a list of page structure names or of page numbers. When either or both lists are not used by the caller, the caller should pass an empty list in the unused one. No lists are needed for P_ALL_PAGES and P_USE_DIALOG operations. A listing of all pages to be printed is required with a P_PAGE_LIST operation, the order of the elements in the list is irrelevant. A list of two pages is needed for the P_PAGE_RANGE operation: the starting and ending page numbers. The lower page number must be the first element in the list.

Arguments

opt Type of print operation desired:

- P_ALL_PAGES - all pages with no dialog
- P_PAGE_LIST - list page structure names
- P_PAGE_RANGE - start and end page numbers
- P_USE_DIALOG - use the dialog

pages List of page IDs of the pages to be printed, or NULL if no list is specified.

nums List of page numbers to be printed, or NULL if no list is specified.

Graphical Functions

Return value

None. The print is executed.

Exception

Raised if unsuccessful.

DSUtil_WorldToPoints

Converts from model coordinates to points (72 to the inch).

Synopsis

```
DSUtil_WorldToPoints : int -> int
```

Description

This routine converts from world units to points (72 to the inch).

Arguments

Value in model units.

Return value

Returns equivalent in points, rounding up a half point or greater.

