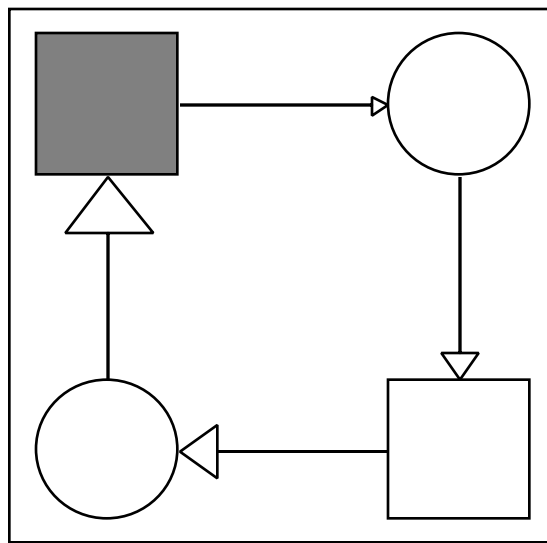


# Design/CPN Internal Functions Programmer's Reference

Version 2.0



## Meta Software Corporation



125 CambridgePark Drive  
Cambridge, MA 02140 U.S.A.

Tel: (617) 576-6920

Fax: (617) 661-2008

© 1993 Meta Software

© 1993 Meta Software Corporation

125 CambridgePark Drive

Cambridge, MA 02140

(617) 576-6920

FAX: (617) 661-2008

email: [cpn-tech-support@metasoft.com](mailto:cpn-tech-support@metasoft.com)

Design/CPN is a trademark of Meta Software Corporation.

# Design/CPN Internal Functions Programmer's Reference

Version 2.0

## Table of Contents

### Part 1: Graphical Functions

#### Chapter 1

##### Symbolic Constants

|   |     |
|---|-----|
| Color Table Indices.....  | 1-2 |
| Integer constants which index into the color table              |     |
| Connector Tip Constants.....                                    | 1-3 |
| Constants to use for determining/setting connector orientations |     |
| Miscellaneous Constants.....                                    | 1-4 |
| Integer constants for use with a variety of functions           |     |
| Node and Connector Shapes.....                                  | 1-5 |
| Symbolic integer constants for node shapes                      |     |
| Node Types.....   | 1-6 |
| The different types of nodes as integer constants               |     |
| Object Flags.....   | 1-7 |
| Integer flags used to read and set properties of objects        |     |
| Object Types.....   | 1-8 |
| The different types of objects as integer constants             |     |
| Print Options.....  | 1-9 |
| Constants for use with printing commands                        |     |

## Chapter 1 Symbolic Constants (cont'd)

|   |      |
|---|------|
| Text Fonts.....   | 1-10 |
| Integer constants for identifying various text fonts              |      |
| Text Justification.....   | 1-11 |
| Integer constants for identifying the various text justifications |      |
| Text Styles.....  | 1-12 |
| Integer constants for identifying the various text styles         |      |

## Chapter 2 Functions for Accessing Graphical Structure

|  |      |
|--|------|
| DSStr_AttachPageToNode.....                            | 2-2  |
| Attaches a subpage to a node and its environment.      |      |
| DSStr_ClosePage.....                                   | 2-3  |
| Closes the given page, if it is open.                  |      |
| DSStr_Coarsen.....                                     | 2-4  |
| Performs a coarsening of the specified page.           |      |
| DSStr_ConnSubGraph.....                                | 2-5  |
| Calculates the set of nodes in a connected subgraph    |      |
| DSStr_CreateConn.....                                  | 2-6  |
| Creates a new connector between the specified nodes.   |      |
| DSStr_CreateLabel.....                                 | 2-7  |
| Creates a new label at position (x,y).                 |      |
| DSStr_CreateLine.....                                  | 2-8  |
| Creates a new line between points.                     |      |
| DSStr_CreateNode.....                                  | 2-9  |
| Creates a new node.                                    |      |
| DSStr_CreatePolygon.....                               | 2-10 |
| Creates a new polygon node.                            |      |
| DSStr_DeleteObject.....                                | 2-11 |
| Deletes the designated object from the model structure |      |
| DSStr_GetConnOtherEnd.....                             | 2-12 |
| Gets the other end of the specified connector.         |      |

## Chapter 2

### Functions for Accessing Graphical Structure (cont'd)

|  |      |
|--|------|
| DSStr_GetCurGroup.....   | 2-13 |
| Gets the members of the current group.                         |      |
| DSStr_GetCurObject.....  | 2-14 |
| Reads the ID of the currently selected object.                 |      |
| DSStr_GetCurPage.....  | 2-15 |
| Reads the ID of the currently selected page.                   |      |
| DSStr_GetDocId.....  | 2-16 |
| Gets the identification number of the current diagram.         |      |
| DSStr_GetInternalConnList.....                                 | 2-17 |
| Gets all the connectors between a set of specified nodes.      |      |
| DSStr_GetNodeList.....   | 2-18 |
| Returns a list of nodes in a specified page.                   |      |
| DSStr_GetObjectConnList.....                                   | 2-19 |
| Determines the connectors that are attached to an object.      |      |
| DSStr_GetObjectInOutLists.....                                 | 2-20 |
| Determines the nodes that are input and output to it.          |      |
| DSStr_GetObjectRegionList.....                                 | 2-21 |
| Determines the IDs of regions in a parent.                     |      |
| DSStr_GetPageConnList.....                                     | 2-22 |
| Determines the IDs of connectors in a page.                    |      |
| DSStr_GetPageList.....   | 2-23 |
| Determines the IDs of pages in a document.                     |      |
| DSStr_GetParent.....   | 2-24 |
| Identifies the parent ID of a page, node, connector or region. |      |
| DSStr_GetTopParent.....  | 2-25 |
| Finds the node or connector parent of a region.                |      |
| DSStr_IsPageOpen.....  | 2-26 |
| Tests whether a page is open or closed.                        |      |
| DSStr_IsValidObject.....                                       | 2-27 |
| Determines whether an object ID represents a valid object.     |      |
| DSStr_MakeNodeIntoRgn.....                                     | 2-28 |
| Changes designated object into a region of designated parent.  |      |

## Chapter 2

### Functions for Accessing Graphical Structure (cont'd)

|   |      |
|---|------|
| DSStr_MakeRgnIntoNode.....                              | 2-29 |
| Calls kernel function to make a region into a node.     |      |
| DSStr_MoveNodesToPage.....                              | 2-30 |
| Moves a set of nodes to a new page.                     |      |
| DSStr_NewPage.....                                      | 2-31 |
| Creates a new page in the document.                     |      |
| DSStr_NewPageWithFlags.....                             | 2-32 |
| Creates a new page in the document.                     |      |
| DSStr_PortNodesOnPage.....                              | 2-33 |
| Returns a list of port nodes on a given page.           |      |
| DSStr_SetCurGroup.....                                  | 2-34 |
| Sets the current group, turning group mode if necessary |      |
| DSStr_SetCurObject.....                                 | 2-35 |
| Changes the current object.                             |      |
| DSStr_SetCurPage.....                                   | 2-36 |
| Sets the current page.                                  |      |
| DSStr_SetDiagModified.....                              | 2-37 |
| Sets the kernel Modified to TRUE or FALSE.              |      |

## Chapter 3

### Functions for Reading Attributes

|   |     |
|---|-----|
| DSFile_GetCurrentDiagName.....                                      | 3-2 |
| Gets the current diagram name.                                      |     |
| DSFile_NameDialog.....  | 3-3 |
| Puts up a file selection dialog to obtain a filename from the user. |     |
| DSRdAttr_ArrowHeadType.....   | 3-4 |
| Returns the connector head type                                     |     |
| DSRdAttr_ConnOrient.....  | 3-5 |
| Reads the orientation information for a connector.                  |     |
| DSRdAttr_ConnPoints.....  | 3-6 |
| Reads the points of a connector of type STRAIGHTCONN.               |     |

**Chapter 3****Functions for Reading Attributes (cont'd)**

|   |      |
|---|------|
| DSRdAttr_ConnProps.....   | 3-7  |
| Reads attributes for a connector, or the global attributes.       |      |
| DSRdAttr_GetConnEnds.....   | 3-8  |
| Reads the two node names for a connector.                         |      |
| DSRdAttr_GetMaxGroupSize.....                                     | 3-9  |
| Returns the maximum allowable size for an aggregate.              |      |
| DSRdAttr_GetObjectCenter.....                                     | 3-10 |
| Finds the current center coordinates of a page, node, or region.  |      |
| DSRdAttr_GetObjectFlags.....                                      | 3-11 |
| Allows the user to read various object flags.                     |      |
| DSRdAttr_GetObjectSize.....                                       | 3-12 |
| Finds the current width and height of a page, node, or region.    |      |
| DSRdAttr_GetObjectSubpage.....                                    | 3-13 |
| Gets the subpage ID if the object is a coarse object.             |      |
| DSRdAttr_GetObjectType.....                                       | 3-14 |
| Reads the object type information associated with an object's ID. |      |
| DSRdAttr_GetOwnedValue.....                                       | 3-15 |
| Reads the node's owned information.                               |      |
| DSRdAttr_GetPageAttr.....   | 3-16 |
| Gets page (field) attributes.                                     |      |
| DSRdAttr_GetParentNode.....                                       | 3-17 |
| Reads the parent information associated with a page.              |      |
| DSRdAttr_GetRegionId.....   | 3-18 |
| Reads the region identifier associated with a region.             |      |
| DSRdAttr_GetRepObject.....  | 3-19 |
| Reads the representative node or connector information.           |      |
| DSRdAttr_GetShape.....  | 3-20 |
| Reads the shape information associated with a structure block.    |      |
| DSRdAttr_GetTextDefaults.....                                     | 3-21 |
| Allows user to read default text attributes.                      |      |
| DSRdAttr_GetType.....   | 3-22 |
| Reads the node type information associated with a node.           |      |

## Chapter 3

### Functions for Reading Attributes (cont'd)

|   |      |
|---|------|
| DSRdAttr_InGroupMode.....                                     | 3-23 |
| Reads the global variable for Group Mode.                     |      |
| DSRdAttr_NetElementType.....                                  | 3-24 |
| Reads or computes node type information.                      |      |
| DSRdAttr_ObjectVisuals.....                                   | 3-25 |
| Reads the attributes of an object that affect its appearance. |      |
| DSRdAttr_PageScale.....                                       | 3-26 |
| Reads the horizontal and vertical scale of a page.            |      |
| DSRdAttr_PolyDefaults.....                                    | 3-27 |
| Allows the user to read the attributes of regular polygons.   |      |
| DSRdAttr_PolyPointCount.....                                  | 3-28 |
| Reads the number of points in a POLYGON or REGPOLY.           |      |
| DSRdAttr_PolyPoints.....                                      | 3-29 |
| Reads the points of a POLYGON or REGPOLY.                     |      |
| DSRdAttr_SegmentCurvature.....                                | 3-30 |
| Returns the segment vertex curvature value for a connector.   |      |
| DSRdAttr_SelectableFlag.....                                  | 3-31 |
| Reads the current value of the Global Selectable flag.        |      |
| DSRdAttr_TextPointSize.....                                   | 3-32 |
| This function reads the text point size of an object.         |      |

## Chapter 4

### Functions for Writing Attributes

|  |     |
|--|-----|
| DSWtAttr_AdjustObjectSize.....                               | 4-2 |
| Changes the width and height of a page, node or region.      |     |
| DSWtAttr_ConnCurvature.....                                  | 4-3 |
| Writes the curvature value for the given connector.          |     |
| DSWtAttr_ConnEndIds.....                                     | 4-4 |
| Changes the connector ends information for a connector.      |     |
| DSWtAttr_ConnOrient.....                                     | 4-5 |
| Changes the orientation info for a connector, or the default |     |



## Chapter 4

### Functions for Writing Attributes (cont'd)

|  |      |
|--|------|
| DSWtAttr_ConnVisuals.....  | 4-6  |
| Writes attributes for a given connector, or the global attributes. |      |
| DSWtAttr_LineThickness.....  | 4-7  |
| Allows the user to control thickness of lines.                     |      |
| DSWtAttr_LineType.....   | 4-8  |
| Allows the user to control line type of an object boundary.        |      |
| DSWtAttr_ObjectFillType.....                                       | 4-9  |
| Allows the user to control fill type of an object or arrowhead.    |      |
| DSWtAttr_ObjectFlags.....  | 4-10 |
| Allows the user to write various object flags.                     |      |
| DSWtAttr_ObjectPosition.....                                       | 4-12 |
| Moves the designated node or region to coordinate position (x,y).  |      |
| DSWtAttr_ObjectVisuals.....  | 4-13 |
| Sets the attributes of an object that affect its appearance.       |      |
| DSWtAttr_PageInfo.....   | 4-15 |
| Changes page attributes.   |      |
| DSWtAttr_RegionId.....   | 4-16 |
| Writes a user-designated region type for a region.                 |      |
| DSWtAttr_RegularPolyInfo.....                                      | 4-17 |
| Allows the user to write the attributes of regular polygons.       |      |
| DSWtAttr_RepNodeId.....  | 4-18 |
| Writes the represented node or represented connector.              |      |
| DSWtAttr_SetConnPoints.....  | 4-19 |
| Writes the points of a connector of type STRAIGHTCONN.             |      |
| DSWtAttr_SetDefaultSelectable.....                                 | 4-20 |
| Writes the current value of the global Selectable flag.            |      |
| DSWtAttr_SetPetriNodeType.....                                     | 4-21 |
| Writes the Petri node type information associated with a node.     |      |
| DSWtAttr_SetPolyPoints.....  | 4-22 |
| Writes the points of a POLYGON or REGPOLY.                         |      |

## Chapter 5 Text Functions

|  |      |
|--|------|
| DSText_Append.....   | 5-2  |
| Adds.append text to an object.                                     |      |
| DSText_Get.....  | 5-3  |
| Reads text from the Text Edit record associated with an object ID. |      |
| DSText_GetLength.....  | 5-4  |
| Determines length of text associated with an object ID.            |      |
| DSText_GetTextParent.....  | 5-5  |
| Returns the text parent of a node, connector, or region.           |      |
| DSText_IsModeOn.....   | 5-6  |
| Reads the current text state.                                      |      |
| DSText_MaxLineLength.....  | 5-7  |
| Returns the width of the longest line of text of a text record.    |      |
| DSText_Put.....  | 5-8  |
| Writes the supplied text into a Text record.                       |      |
| DSText_SetAttr.....  | 5-9  |
| Allows user to write text attributes for a given object.           |      |
| DSText_SetDefaultFont.....   | 5-10 |
| Changes the default font.  |      |
| DSText_SetDefaultJust.....   | 5-11 |
| Changes the default text justification.                            |      |
| DSText_SetDefaultSize.....   | 5-12 |
| Changes the default point size.                                    |      |
| DSText_SetDefaultStyle.....  | 5-13 |
| Changes the default text style.                                    |      |
| DSText_SetMode.....  | 5-14 |
| Writes the current text state.                                     |      |

## Chapter 6 User Interface Functions

|   |     |
|---|-----|
| DSUI_Align.....                                 | 6-2 |
| Aligns an object with respect to other objects. |     |

## Chapter 6 (cont'd) User Interface Functions

|  |      |
|--|------|
| DSUI_AskUserToSelectPage.....                                    | 6-3  |
| Prompts user to select a page.                                   |      |
| DSUI_AutoPan.....  | 6-4  |
| Pans the current page to make the specified object visible.      |      |
| DSUI_BeepUser.....   | 6-5  |
| Makes the machine BEEP.  |      |
| DSUI_ChangeCursor.....   | 6-6  |
| Changes the appearance of the cursor.                            |      |
| DSUI_CheckBounds.....  | 6-7  |
| Checks the user's desired value of a dialog item.                |      |
| DSUI_Cleanup.....  | 6-8  |
| Cleans up the graphical structure of specified page.             |      |
| DSUI_Duplicate.....  | 6-9  |
| Duplicates a given set of nodes on a given page.                 |      |
| DSUI_GetIntegerValue.....  | 6-10 |
| To prompt the user for an integer value.                         |      |
| DSUI_GetString.....  | 6-11 |
| To prompt the user for a string.                                 |      |
| DSUI_GetUserYesOrNo.....   | 6-12 |
| To prompt the user to make a two-way decision.                   |      |
| DSUI_Indicate.....   | 6-13 |
| Indicates a group of nodes and/or regions on the screen.         |      |
| DSUI_IndicateObject.....   | 6-14 |
| Draws the dot handles for the specified object.                  |      |
| DSUI_MakePageVisible.....  | 6-15 |
| Makes a page visible if is not currently visible.                |      |
| DSUI_Merge.....  | 6-16 |
| Merges a group of nodes into a target node.                      |      |
| DSUI_NoUndo.....   | 6-17 |
| To prevent Undo operations.                                      |      |
| DSUI_PreventObjectAdjust.....                                    | 6-18 |
| To prevent user from performing size adjustments on all objects. |      |

### **Chapter 6 (cont'd)** **User Interface Functions**

|   |      |
|---|------|
| DSUI_Redraw.....  | 6-19 |
| Redraws the specified object.                               |      |
| DSUI_RestoreStatusBar.....                                  | 6-20 |
| Clears any message in the status bar.                       |      |
| DSUI_SelectObject.....                                      | 6-21 |
| Prompts the user to select an object.                       |      |
| DSUI_SetObjectIndication.....                               | 6-22 |
| To enable/disable the indicate dot feature for all objects. |      |
| DSUI_SetRepConnDeleteMode.....                              | 6-23 |
| Sets the treatment of represented connector deletion.       |      |
| DSUI_SetStatusBarMessage.....                               | 6-24 |
| Puts a message in the status bar.                           |      |
| DSUI_Spread.....  | 6-25 |
| Spreads a grouping of three or more nodes.                  |      |
| DSUI_UpdateCurrentPage.....                                 | 6-26 |
| Redraws the current page on the screen.                     |      |
| DSUI_UserAckMessage.....                                    | 6-27 |
| Displays user supplied message as a modal dialog.           |      |

### **Chapter 7** **Utility Functions**

|  |     |
|--|-----|
| DSUtil_DrawArc.....  | 7-2 |
| Calculates and draws a circular curve.                         |     |
| DSUtil_GetConnClipPoint.....                                   | 7-3 |
| Gets a clip point and puts it directly into the points vector. |     |
| DSUtil_IsALabel.....   | 7-4 |
| Tells whether an object is a label.                            |     |
| DSUtil_LineToInCoords.....                                     | 7-5 |
| Draws a line from the current pen position to a given point.   |     |
| DSUtil_Pause.....  | 7-6 |
| Pauses the specified number of time units.                     |     |

### **Chapter 7 (cont'd)** **Utility Functions**

|   |      |
|---|------|
| DSUtil_PointInObject.....                                       | 7-7  |
| Determines if the given point is inside a given object type.    |      |
| DSUtil_PointsToWorld.....                                       | 7-8  |
| Converts from points (72 to the inch) to model units.           |      |
| DSUtil_PrintPages.....  | 7-9  |
| Prints any set of pages with or without using the print dialog. |      |
| DSUtil_WorldToPoints.....                                       | 7-11 |
| Converts from model coordinates to points (72 to the inch).     |      |

## Part 2: Diagram Functions

### Chapter 8

#### Converting Auxiliary Objects to CPN Objects

|   |      |
|---|------|
| MakeCpnPage.....  | 8-2  |
| Converts a newly created Design page into a CPN page.         |      |
| MakeGlobDec.....  | 8-3  |
| Converts an auxiliary node into a CPN global declaration.     |      |
| MakeTempDec.....  | 8-4  |
| Converts an auxiliary node into a CPN temporary declaration.  |      |
| MakeLocDec.....   | 8-5  |
| Converts an auxiliary node into a CPN local declaration.      |      |
| MakePlace.....  | 8-6  |
| Converts an auxiliary node into a CPN place.                  |      |
| MakeTrans.....  | 8-7  |
| Converts an auxiliary node into a CPN transition.             |      |
| MakeArc.....  | 8-8  |
| Converts an existing connector into a CPN arc.                |      |
| MakeName.....   | 8-9  |
| Converts an auxiliary node into a CPN name region.            |      |
| MakeColor.....  | 8-10 |
| Converts an auxiliary node into a CPN color set region.       |      |
| MakeInitMark.....   | 8-11 |
| Converts an auxiliary node into a CPN initial marking region. |      |
| MakeGuard.....  | 8-12 |
| Converts an auxiliary node into a CPN guard region.           |      |
| MakeCodeSeg.....  | 8-13 |
| Converts an auxiliary node into a CPN code segment region.    |      |
| MakeTime.....   | 8-14 |
| Converts an auxiliary node into a CPN time region.            |      |
| MakePrimePage.....  | 8-15 |
| Flags a CPN page node as a prime page.                        |      |
| MakeArcExp.....   | 8-16 |
| Converts an auxiliary node into a CPN arc expression region.  |      |

## Chapter 9 Creating CPN Hierarchies

|   |      |
|---|------|
| MakeInPort.....   | 9-2  |
| Converts a CPN place into a CPN input port node.              |      |
| MakeOutPort.....  | 9-3  |
| Converts a CPN place into a CPN output port node.             |      |
| MakeInOutPort.....  | 9-4  |
| Converts a CPN place into a CPN input/output port.            |      |
| MakeGenPort.....  | 9-5  |
| Converts a CPN place into a CPN general port.                 |      |
| MakeTransSub.....   | 9-6  |
| Converts a CPN transition into a CPN substitution transition. |      |
| AssignPort.....   | 9-7  |
| Assigns a CPN port node to a CPN socket node.                 |      |
| MakeGlobalPlaceFus.....                                       | 9-8  |
| Creates a CPN global fusion set and adds a CPN place to it.   |      |
| MakePagePlaceFus.....   | 9-9  |
| Creates a CPN page fusion set and adds a CPN place to it.     |      |
| MakeInstPlaceFus.....   | 9-10 |
| Creates a CPN instance fusion set and adds a CPN place to it. |      |
| AddToGlobalPlaceFus.....                                      | 9-11 |
| Adds a CPN place to a CPN global fusion set.                  |      |
| AddToPagePlaceFus.....  | 9-12 |
| Adds a CPN place to a CPN page or instance fusion set.        |      |
| AddToGlobalTransFus.....                                      | 9-13 |
| Adds a CPN transition to a CPN global fusion set.             |      |
| AddToPageTransFus.....  | 9-14 |
| Adds a CPN transition to a CPN page or instance fusion set.   |      |

## Chapter 10 Accessing CPN Diagram Structure

|   |      |
|---|------|
| IsGlobalPlaceFus.....                                     | 10-2 |
| Verifies if a string is used as a global fusion set name. |      |

### Chapter 10 (cont'd)

#### Accessing CPN Diagram Structure

|   |       |
|---|-------|
| IsPagePlaceFus.....   | 10-3  |
| Verifies if a string is used as a page or instance fusion set name. |       |
| IsPort.....   | 10-4  |
| Verifies if a CPN place or transition is a CPN port node.           |       |
| GetPageName.....  | 10-5  |
| Gets the text of the name of a CPN page.                            |       |
| GetNameText.....  | 10-6  |
| Gets the text of the name region of a CPN place or transition.      |       |
| GetName.....  | 10-7  |
| Gets the ID of the CPN name region of a CPN place or transition.    |       |
| GetColor.....   | 10-8  |
| Gets the ID of the CPN color set region of a CPN place.             |       |
| GetInitMark.....  | 10-9  |
| Gets the ID of the CPN initial marking region of a CPN place.       |       |
| GetGuard.....   | 10-10 |
| Gets the ID of the CPN guard region of a CPN transition.            |       |
| GetTime.....  | 10-11 |
| Gets the ID of the CPN time region of a CPN transition.             |       |
| GetCodeSeg.....   | 10-12 |
| Gets the ID of the code segment region of a node.                   |       |
| GetArcExp.....  | 10-13 |
| Gets the ID of the CPN arc expression region of a CPN arc.          |       |
| GetPort.....  | 10-14 |
| Gets the ID of the CPN port region of a CPN place or transition.    |       |
| GetFusion.....  | 10-15 |
| Gets the ID of the CPN fusion set region of a place.                |       |
| GetCpnInfo.....   | 10-16 |
| Gets information on CPN objects.                                    |       |



## Chapter 11 Obtaining CPN Simulation Information

|   |       |
|---|-------|
| IsPageIncluded.....   | 11-2  |
| Verifies if a CPN page is included in the current simulation.       |       |
| IsPageProposed.....   | 11-3  |
| Verifies if a page participates in occurrence set calculation.      |       |
| IsPageObserv.....   | 11-4  |
| Verifies if a CPN page is observable in the current simulation.     |       |
| IsPageCode.....   | 11-5  |
| Verifies if the transitions of a page execute with code segments.   |       |
| IsPageAuto.....   | 11-6  |
| Verifies if a CPN page participates in an automatic run.            |       |
| GetPageModeAttr.....  | 11-7  |
| Gets the simulation mode attributes for a CPN page.                 |       |
| IsSubTransIncluded.....   | 11-8  |
| Verifies if a subpage is included in the current simulation.        |       |
| IsSubTransProposed.....   | 11-9  |
| Verifies if a subpage participates in occurrence set calculation.   |       |
| IsSubTransObserv.....   | 11-10 |
| Verifies if the subpage of a substitution transition is observable. |       |
| IsSubTransCode.....   | 11-11 |
| Verifies if transitions of a subpage execute with code segments.    |       |
| IsSubTransAuto.....   | 11-12 |
| Verifies if a subpage participates in an automatic run.             |       |
| GetSubTransModeAttr.....  | 11-13 |
| Gets the simulation mode attributes for a subpage.                  |       |
| IsPagePrime.....  | 11-14 |
| Verifies if a CPN page is a prime page.                             |       |
| GetPageMult.....  | 11-15 |
| Gets the multiplicity of a CPN page.                                |       |
| GetPageInsts.....   | 11-16 |
| Gets the IDs of CPN page instances.                                 |       |
| GetPageInstName.....  | 11-17 |
| Gets the name of a CPN page instance.                               |       |

### **Chapter 11 (cont'd)**

#### **Obtaining CPN Simulation Information**

|  |       |
|--|-------|
| GetPageInstComp.....                                       | 11-18 |
| Gets the ID of a CPN page instance compound node.          |       |
| GetMarkingCode.....  | 11-19 |
| Gets the ML code for accessing the marking of a CPN place. |       |
| GetChangeMarkingCode.....                                  | 11-20 |
| Gets the ML code for changing the marking of a CPN place.  |       |

---

## Part 3: Reporting Functions

### Chapter 12 Statistical Variable Functions

|   |       |
|---|-------|
| SV'avrg.....  | 12-2  |
| Returns the average value of a statistical variable.                |       |
| SV'count.....   | 12-3  |
| Returns the number of times SV'upd has been called                  |       |
| SV'createint.....   | 12-5  |
| Creates an integer statistical variable.                            |       |
| SV'first.....   | 12-6  |
| Returns the first value of a statistical variable.                  |       |
| SV'init.....  | 12-7  |
| Initializes a statistical variable.                                 |       |
| SV'max.....   | 12-8  |
| Returns the max of the values of a statistical variable.            |       |
| SV'min.....   | 12-9  |
| Returns the min of the values of a statistical variable.            |       |
| SV'ss.....  | 12-10 |
| Returns the sum of squares of the values of a statistical variable. |       |
| SV'ssd.....   | 12-11 |
| Returns the sum of squares of deviation of a statistical variable.  |       |
| SV'std.....   | 12-12 |
| Returns the standard deviation of a statistical variable.           |       |
| SV'sum.....   | 12-13 |
| Returns the sum of the values of a statistical variable.            |       |
| SV'upd.....   | 12-14 |
| Updates a statistical variable.                                     |       |
| SV'value.....   | 12-15 |
| Returns the current value of a statistical variable.                |       |
| SV'vari.....  | 12-16 |
| Returns the variance of a statistical variable.                     |       |

## Chapter 13 Overview of Chart Functions

|                               |      |
|-------------------------------|------|
| Table of Chart Functions..... | 13-1 |
|-------------------------------|------|

## Chapter 14 Bar Chart Functions

|  |       |
|--|-------|
| BC_create.....   | 14-3  |
| Creates a bar chart.                                       |       |
| BC_clear_chart.....  | 14-9  |
| Clears a bar chart.  |       |
| BC_delete.....   | 14-10 |
| Deletes a bar chart.                                       |       |
| BC_GetCodeSeg.....   | 14-11 |
| Gets the ID of the CPN code segment region of a bar chart. |       |
| BC_init_chart.....   | 14-12 |
| Initializes a bar chart.                                   |       |
| BC_upd_partnames.....                                      | 14-13 |
| Updates the partnames in a bar chart.                      |       |
| BC_upd_title.....  | 14-14 |
| Updates the title of a bar chart.                          |       |
| HC_upd_bar.....  | 14-15 |
| Updates the current bar of a history bar chart.            |       |
| HC_upd_barnames.....                                       | 14-16 |
| Updates the barnames in a history bar chart.               |       |
| HC_upd_chart.....  | 14-17 |
| Updates bars in a history bar chart.                       |       |
| SC_upd_bar.....  | 14-18 |
| Updates a bar of a snapshot bar chart.                     |       |
| SC_upd_barnames.....                                       | 14-19 |
| Updates the barnames in a snapshot bar chart.              |       |
| SC_upd_chart.....  | 14-20 |
| Updates bars in a snapshot bar chart.                      |       |
| SC_upd_part.....   | 14-21 |
| Updates a part for the bars of a snapshot bar chart.       |       |

## Chapter 15 Line Chart Functions

|   |       |
|---|-------|
| LC_create.....  | 15-3  |
| Creates a line chart.   |       |
| LC_delete.....  | 15-8  |
| Deletes a line chart.   |       |
| LC_GetCodeSeg.....  | 15-9  |
| Gets the ID of the CPN code segment region of a line chart.     |       |
| LC_init_chart.....  | 15-10 |
| Initializes a line chart.                                       |       |
| LC_upd_axisnames.....   | 15-11 |
| Updates the axis names in a line chart.                         |       |
| LC_upd_chart.....   | 15-12 |
| Updates lines in a line chart.                                  |       |
| LC_upd_line.....  | 15-13 |
| Updates a line of a line chart.                                 |       |
| LC_upd_linenames.....   | 15-14 |
| Updates the line names in a line chart.                         |       |
| LC_upd_pline.....   | 15-15 |
| Updates the fill pattern for a line terminator in a line chart. |       |
| LC_upd_title.....   | 15-16 |
| Updates the title of a line chart.                              |       |

## Appendix A: Version 1.9 Chart Functions

### Chapter A1

#### Overview of Version 1.9 Chart Functions

|  |      |
|--|------|
| Contents of Appendix A.....                      | A1-1 |
| Obsolescence of Version 1.9 Chart Functions..... | A1-1 |
| Table of Version 1.9 Chart Functions.....        | A1-1 |
| Creating a Chart Page.....                       | A1-2 |
| Positioning a Chart on a Page.....               | A1-2 |

### Chapter A2

#### Version 1.9 Bar Chart Functions

|  |       |
|--|-------|
| BC'create.....                                 | A2-2  |
| Creates a bar chart.                           |       |
| BC'decint.....                                 | A2-6  |
| Declares an integer bar chart.                 |       |
| BC'delete.....                                 | A2-7  |
| Deletes an integer bar chart.                  |       |
| BC'upd_col.....                                | A2-8  |
| Updates columns in an integer bar chart.       |       |
| BC'upd_ltag.....                               | A2-10 |
| Updates legend labels on an integer bar chart. |       |
| BC'upd_row.....                                | A2-11 |
| Updates rows in an integer bar chart.          |       |
| BC'upd_rtag.....                               | A2-13 |
| Updates row labels on an integer bar chart.    |       |

### Chapter A3

#### Version 1.9 History Chart Functions

|  |      |
|--|------|
| HC'create.....                         | A3-2 |
| Creates a history bar chart.           |      |
| HC'decint.....                         | A3-6 |
| Declares an integer history bar chart. |      |
| HC'delete.....                         | A3-7 |
| Deletes an integer history bar chart.  |      |

## Chapter A3 (cont'd)

### Version 1.9 History Chart Functions

|   |       |
|---|-------|
| HC'hist_ltag.....   | A3-8  |
| Updates column legend labels on an integer history bar chart. |       |
| HC'hist_init.....   | A3-9  |
| Initializes columns in an integer history bar chart.          |       |
| HC'hist_row.....  | A3-11 |
| Creates a new row in an integer history bar chart.            |       |
| HC'hist_rtag.....   | A3-13 |
| Updates row labels on an integer history bar chart.           |       |

## Chapter A4

### Version 1.9 Line Graph Functions

|   |       |
|---|-------|
| LG'create.....                                  | A4-2  |
| Creates a line graph.                           |       |
| LG'decint.....                                  | A4-6  |
| Declares an integer line graph.                 |       |
| LG'delete.....                                  | A4-7  |
| Deletes an integer line graph.                  |       |
| LG'upd_atag.....                                | A4-8  |
| Updates axis labels for an integer line graph.  |       |
| LG'upd_ltag.....                                | A4-9  |
| Updates legend labels on an integer line graph. |       |
| LG'upd_line.....                                | A4-10 |
| Updates an integer line graph.                  |       |
| LG'upd_pline.....                               | A4-12 |
| Updates the pattern of an integer line graph.   |       |

## Chapter A5

### Version 1.9 Matrix Chart Functions

|                                   |      |
|-----------------------------------|------|
| MC'create.....                    | A5-2 |
| Creates a matrix chart.           |      |
| MC'dec.....                       | A5-5 |
| Declares an integer matrix chart. |      |

### Chapter A5 (cont'd)

#### Version 1.9 Matrix Chart Functions

|   |       |
|---|-------|
| MC'create.....  | A5-2  |
| Creates a matrix chart.                                   |       |
| MC'dec.....   | A5-5  |
| Declares an integer matrix chart.                         |       |
| MC'delete.....  | A5-6  |
| Deletes an integer matrix chart.                          |       |
| MC'fill.....  | A5-7  |
| Updates the fill pattern in an integer matrix chart.      |       |
| MC'upd_ltag.....  | A5-9  |
| Updates pattern legend labels on an integer matrix chart. |       |
| MC'write.....   | A5-10 |
| Updates cell values in an integer matrix chart.           |       |



# INDEX



## A

**AddToGlobalPlaceFus,**

Adds a CPN place to a CPN global fusion set; 9-11

**AddToGlobalTransFus,**

Adds a CPN transition to a CPN global fusion set; 9-13

**AddToPagePlaceFus,**

Adds a CPN place to a CPN page or instance fusion set; 9-12

**AddToPageTransFus,**

Adds a CPN transition to a CPN page or instance fusion set; 9-14

**AssignPort,**

Assigns a CPN port node to a CPN socket node for a CPN compound node; 9-7

## B

**BC'create,**

Creates a bar chart; A2-2

**BC'decint,**

Declares an integer bar chart; A2-6

**BC'delete,**

Deletes an integer bar chart; A2-7

**BC'upd\_col,**

Updates columns in an integer bar chart; A2-8

**BC'upd\_ltag,**

Updates legend labels on an integer bar chart; A2-10

**BC'upd\_row,**

Updates rows in an integer bar chart; A2-11

**BC'upd\_rtag,**

Updates row labels on an integer bar chart; A2-13

**BC\_clear\_chart,**

Clears a bar chart; 14-9

**BC\_create,**

Creates a bar chart; 14-3

**BC\_delete,**

Deletes a bar chart; 14-10

**BC\_GetCodeSeg,**

Gets the ID of the CPN code segment region of a bar chart; 14-11

**BC\_init\_chart,**

Initializes a bar chart; 14-12

**BC\_upd\_partnames,**

Updates the partnames in a bar chart.; 14-13

**BC\_upd\_title,**

Updates the title of a bar chart; 14-14

## C

**Color Table Indices,**

Integer constants which index into the color table; 1-2

**Connector Tip Constants,**

Integer constants to use for determining/setting connector orientations; 1-3

## D

**DSFile\_GetCurrentDiagName,**

Gets the current diagram name; 3-2

**DSFile\_NameDialog,**

Puts up a file selection dialog to obtain a filename from the user; 3-3

**DSRdAttr\_ArrowHeadType,**

Returns the connector head type; 3-4

**DSRdAttr\_ConnOrient,**

Reads the orientation information for a connector; 3-5

**DSRdAttr\_ConnPoints,**

Reads the points of a connector of type STRAIGHTCONN; 3-6

**DSRdAttr\_ConnProps,**

Reads attributes for a given connector, or reads the global attributes for connector creation; 3-7

**DSRdAttr\_GetConnEnds,**

Reads the two node names for a connector; 3-8

**DSRdAttr\_GetMaxGroupSize,**

Returns the maximum allowable size for an aggregate; 3-9

**DSRdAttr\_GetObjectCenter,**

Finds the current center coordinates of a page, node, or region; 3-10

**DSRdAttr\_GetObjectFlags,**

Allows the user to read various object flags; 3-11

**DSRdAttr\_GetObjectSize,**

Finds the current width and height of a page, node, or region; 3-12

**DSRdAttr\_GetObjectSubpage,**

Gets the subpage ID if the object is a coarse object (node or connector); 3-13

**DSRdAttr\_GetObjectType,**

Reads the object type information associated with an object's ID; 3-14

**DSRdAttr\_GetOwnedValue,**

Reads the node's owned information; 3-15

# Index

---

- DSRdAttr\_GetPageAttr,**
  - Gets page (field) attributes; 3-16
- DSRdAttr\_GetParentNode,**
  - Reads the parent information associated with a page; 3-17
- DSRdAttr\_GetRegionId,**
  - Reads the region identifier associated with a region; 3-18
- DSRdAttr\_GetRepObject,**
  - Reads the representative node or representative connector information for a node or connector, respectively; 3-19
- DSRdAttr\_GetShape,**
  - Reads the object shape information associated with a structure block in the model; 3-20
- DSRdAttr\_GetTextDefaults,**
  - Allows user to read default text attributes; 3-21
- DSRdAttr\_GetType,**
  - Reads the node type information associated with a node; 3-22
- DSRdAttr\_InGroupMode,**
  - Reads the global variable for Group Mode; 3-23
- 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; 3-24
- DSRdAttr\_ObjectVisuals,**
  - Reads the attributes of an object that affect its appearance; 3-25
- DSRdAttr\_PageScale,**
  - Reads the horizontal and vertical scale of a page; 3-26
- DSRdAttr\_PolyDefaults,**
  - Allows the user to read the attributes of regular polygons; 3-27
- DSRdAttr\_PolyPointCount,**
  - Reads the number of points in a POLYGON or REGPOLY; 3-28
- DSRdAttr\_PolyPoints,**
  - Reads the points of a POLYGON or REGPOLY; 3-29
- DSRdAttr\_SegmentCurvature,**
  - Returns the segment vertex curvature value for the given connector; 3-30
- DSRdAttr\_SelectableFlag,**
  - Reads the current value of the Global Selectable flag; 3-31
- DSRdAttr\_TextPointSize,**
  - This function reads the text point size of an object; 3-32
- DSStr\_AttachPageToNode,**
  - Attaches a subpage to a node and its environment; 2-2
- DSStr\_ClosePage,**
  - Closes the given page, if it is open; 2-3
- DSStr\_Coarsen,**
  - Performs a coarsening of the specified page; 2-4
- DSStr\_ConnSubGraph,**
  - Given a node, calculates the set of nodes in the connected subgraph; 2-5
- DSStr\_CreateConn,**
  - Creates a new connector between the specified nodes; 2-6
- DSStr\_CreateLabel,**
  - Creates a new label at position (x,y); 2-7
- DSStr\_CreateLine,**
  - Creates a new line between points; 2-8
- DSStr\_CreateNode,**
  - Creates a new node; 2-9
- DSStr\_CreatePolygon,**
  - Creates a new polygon node; 2-10
- DSStr\_DeleteObject,**
  - Deletes the designated object from the model structure and frees all space occupied; 2-11
- DSStr\_GetConnOtherEnd,**
  - Gets the other end of the specified connector; 2-12
- DSStr\_GetCurGroup,**
  - Gets the members of the current group; 2-13
- DSStr\_GetCurObject,**
  - Reads the ID of the currently selected object; 2-14
- DSStr\_GetCurPage,**
  - Reads the ID of the currently selected page; 2-15
- DSStr\_GetDocId,**
  - Gets the identification number of the current diagram; 2-16
- DSStr\_GetInternalConnList,**
  - Gets all the connectors between a set of specified nodes; 2-17
- DSStr\_GetNodeList,**
  - Returns a list of nodes in a specified page; 2-18
- DSStr\_GetObjectConnList,**
  - Given an object (node or region), determines the connectors that are attached to the object; 2-19
- DSStr\_GetObjectInOutLists,**
  - Given a node, determines the nodes that are input to it and the nodes that are its outputs; 2-20
- DSStr\_GetObjectRegionList,**
  - Determines the IDs of regions in a parent; 2-21
- DSStr\_GetPageConnList,**
  - Determines the IDs of connectors in a page; 2-22
- DSStr\_GetPageList,**
  - Determines the IDs of pages in a document; 2-23
- DSStr\_GetParent,**
  - Identifies the parent ID of a page, node, connector or region; 2-24
- DSStr\_GetTopParent,**
  - Finds the node or connector parent of a region; 2-25

- DSStr\_IsPageOpen,**  
Tests whether a page is open or closed; 2-26
- DSStr\_IsValidObject,**  
Determines whether the object ID represents a valid object in the document; 2-27
- DSStr\_MakeNodeIntoRgn,**  
Changes designated object into a region of designated parent; 2-28
- DSStr\_MakeRgnIntoNode,**  
Calls kernel function to make a region into a node, to unregionize an object; 2-29
- DSStr\_MoveNodesToPage,**  
Moves a set of nodes to a new page; 2-30
- DSStr\_NewPage,**  
Creates a new page in the document; 2-31
- DSStr\_NewPageWithFlags,**  
Creates a new page in the document; 2-32
- DSStr\_PortNodesOnPage,**  
Returns a list of port nodes on a given page; 2-33
- DSStr\_SetCurGroup,**  
Sets the current group, turning group mode on if it is not already on; 2-34
- DSStr\_SetCurObject,**  
Changes the current object; 2-35
- DSStr\_SetCurPage,**  
Sets the current page; 2-36
- DSStr\_SetDiagModified,**  
Sets the kernel Modified to TRUE or FALSE; 2-37
- DSText\_Append,**  
Adds text to an object. If the object already contains text, the new text is appended to it; 5-2
- DSText\_Get,**  
Reads text from the Text Edit record associated with an object ID; 5-3
- DSText\_GetLength,**  
Determines length of text associated with an object ID; 5-4
- DSText\_GetTextParent,**  
Returns the text parent of a node, connector, or region; 5-5
- DSText\_IsModeOn,**  
Reads the current text state; 5-6
- DSText\_MaxLineLength,**  
Returns the width of the longest line of text of a text record; 5-7
- DSText\_Put,**  
Writes the supplied text into the Text record associated with a model ID. Any text currently in the model ID is deleted; 5-8
- DSText\_SetAttr,**  
Allows user to write text attributes for a given object; 5-9
- DSText\_SetDefaultFont,**  
Changes the default font; 5-10
- DSText\_SetDefaultJust,**  
Changes the default text justification; 5-11
- DSText\_SetDefaultSize,**  
Changes the default point size; 5-12
- DSText\_SetDefaultStyle,**  
Changes the default text style; 5-13
- DSText\_SetMode,**  
Writes the current text state; 5-14
- DSUI\_Align,**  
Aligns an object with respect to other objects; 6-2
- DSUI\_AskUserToSelectPage,**  
Prompts user to select a page; 6-3
- DSUI\_AutoPan,**  
Pans the current page to make the specified object visible; 6-4
- DSUI\_BeepUser,**  
Makes the machine BEEP; 6-5
- DSUI\_ChangeCursor,**  
Changes the appearance of the cursor; 6-6
- 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; 6-7
- DSUI\_Cleanup,**  
Cleans up the graphical structure of specified page; 6-8
- DSUI\_Duplicate,**  
Duplicates a given set of nodes on a given page, preserving their positions; 6-9
- DSUI\_GetIntegerValue,**  
To prompt the user for an integer value; 6-10
- DSUI\_GetString,**  
To prompt the user for a string; 6-11
- DSUI\_GetUserYesOrNo,**  
To prompt the user to make a two-way decision; 6-12
- DSUI\_Indicate,**  
Indicates a group of nodes and/or regions on the screen; 6-13
- DSUI\_IndicateObject,**  
Draws the dot handles for the specified object; 6-14
- DSUI\_MakePageVisible,**  
Makes a page visible if is not currently visible; 6-15
- DSUI\_Merge,**  
Merges a group of nodes into a target node; 6-16
- DSUI\_NoUndo,**  
To prevent Undo operations; 6-17
- DSUI\_PreventObjectAdjust,**  
To prevent user from performing size adjustments on all objects; 6-18

# Index

---

**DSUI\_Redraw,**

Redraws the specified object; 6-19

**DSUI\_RestoreStatusBar,**

Clears any message in the status bar, restoring the normal display; 6-20

**DSUI\_SelectObject,**

Prompts the user to select an object; 6-21

**DSUI\_SetObjectIndication,**

To enable/disable the indicate dot feature for all objects; 6-22

**DSUI\_SetRepConnDeleteMode,**

Sets the treatment of represented connector deletion; 6-23

**DSUI\_SetStatusBarMessage,**

Puts a message in the status bar; 6-24

**DSUI\_Spread,**

Spreads a grouping of three or more nodes to achieve equal space between them; 6-25

**DSUI\_UpdateCurrentPage,**

Redraws the current page on the screen; 6-26

**DSUI\_UserAckMessage,**

Displays user supplied message as a modal dialog; 6-27

**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; 7-2

**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; 7-3

**DSUtil\_IsALabel,**

Tells whether an object is a label; 7-4

**DSUtil\_LineToInCoords,**

Draws a line from the current pen position to a given point; 7-5

**DSUtil\_Pause,**

Pauses the specified number of time units. One unit equals 1/60 of a second; 7-6

**DSUtil\_PointInObject,**

Determines if the given point is inside one of the given object types on the current page; 7-7

**DSUtil\_PointsToWorld,**

Converts from points (72 to the inch) to model units; 7-8

**DSUtil\_PrintPages,**

Allows the OA application to print any set of pages with or without using the print dialog; 7-9

**DSUtil\_WorldToPoints,**

Converts from model coordinates to points (72 to the inch); 7-11

**DSWtAttr\_AdjustObjectSize,**

Allows the width and height of a page, node or region to be changed; 4-2

**DSWtAttr\_ConnCurvature,**

Writes the curvature value for the given connector; 4-3

**DSWtAttr\_ConnEndIds,**

Changes the connector ends information for a connector; 4-4

**DSWtAttr\_ConnOrient,**

Changes the orientation information for a connector, or the default connector orientation; 4-5

**DSWtAttr\_ConnVisuals,**

Writes attributes for a given connector, or writes the global attributes for connector creation; 4-6

**DSWtAttr\_LineThickness,**

Allows the user to control thickness of lines; 4-7

**DSWtAttr\_LineType,**

Allows the user to control line type of an object boundary; 4-8

**DSWtAttr\_ObjectFillType,**

Allows the user to control fill type of an object or arrowhead; 4-9

**DSWtAttr\_ObjectFlags,**

Allows the user to write various object flags; 4-10

**DSWtAttr\_ObjectPosition,**

Moves the designated node or region to coordinate position (x,y); 4-12

**DSWtAttr\_ObjectVisuals,**

Sets the attributes of an object that affect its appearance; 4-13

**DSWtAttr\_PageInfo,**

Changes page attributes; 4-15

**DSWtAttr\_RegionId,**

Writes a user-designated region type for a region; 4-16

**DSWtAttr\_RegularPolyInfo,**

Allows the user to write the attributes of regular polygons; 4-17

**DSWtAttr\_RepNodeId,**

Writes the represented node or represented connector information for a node or connector, respectively; 4-18

**DSWtAttr\_SetConnPoints,**

Writes the points of a connector of type STRAIGHTCONN, replacing the old set of points; 4-19

**DSWtAttr\_SetDefaultSelectable,**

Writes the current value of the global Selectable flag; 4-20

**DSWtAttr\_SetPetriNodeType,**

Writes the Petri node type information associated with a node; 4-21

**DSWtAttr\_SetPolyPoints,**

Writes the points of a POLYGON or REGPOLY, replacing the old set of points; 4-22

## G

**GetArcExp,**

Gets the ID of the CPN arc expression region of a CPN arc; 10-13

**GetChangeMarkingCode,**

Gets the ML code for changing the marking of a CPN place on a given CPN page instance; 11-20

**GetCodeSeg,**

Gets the ID of the CPN code segment region of a CPN transition or chart node; 10-12

**GetColor,**

Gets the ID of the CPN color set region of a CPN place; 10-8

**GetCpnInfo,**

Gets information on CPN objects; 10-16

**GetFusion,**

Gets the ID of the CPN fusion set region of a CPN place or transition; 10-15

**GetGuard,**

Gets the ID of the CPN guard region of a CPN transition; 10-10

**GetInitMark,**

Gets the ID of the CPN initial marking region of a CPN place; 10-9

**GetMarkingCode,**

Gets the ML code for accessing the marking of a CPN place on a given CPN page instance; 11-19

**GetName,**

Gets the ID of the CPN name region of a CPN place or transition; 10-7

**GetNameText,**

Gets the text of the name region of a CPN place or transition; 10-6

**GetPageInstComp,**

Gets the ID of a CPN page instance compound node; 11-18

**GetPageInstName,**

Gets the name of a CPN page instance; 11-17

**GetPageInsts,**

Gets the IDs of CPN page instances; 11-16

**GetPageModeAttr,**

Gets the simulation mode attributes for a CPN page; 11-7

**GetPageMult,**

Gets the multiplicity of a CPN page; 11-15

**GetPageName,**

Gets the text of the name of a CPN page; 10-5

**GetPort,**

Gets the ID of the CPN port region of a CPN place or transition; 10-14

**GetSubTransModeAttr,**

Gets the simulation mode attributes for the subpage of a CPN substitution transition; 11-13

**GetTime,**

Gets the ID of the CPN time region of a CPN transition; 10-11

## H

**HC'create,**

Creates a history bar chart; A3-2

**HC'decint,**

Declares an integer history bar chart; A3-6

**HC'delete,**

Deletes an integer history bar chart; A3-7

**HC'hist\_init,**

Initializes columns in an integer history bar chart; A3-9

**HC'hist\_ltag,**

Updates column legend labels on an integer history bar chart; A3-8

**HC'hist\_row,**

Creates a new row in an integer history bar chart; A3-11

**HC'hist\_rtag,**

Updates row labels on an integer history bar chart; A3-13

**HC\_upd\_bar,**

Updates the current bar of a history bar chart; 14-15

**HC\_upd\_barnames,**

Updates the barnames in a history bar chart; 14-16

**HC\_upd\_chart,**

Updates bars in a history bar chart; 14-17

## I

**IsGlobalPlaceFus,**

Verifies if a string of characters is used as a CPN place global fusion set name; 10-2

**IsPageAuto,**

Verifies if a CPN page participates in the automatic run for the current simulation; 11-6

# Index

---

## **IsPageCode,**

Verifies if the CPN transitions of a CPN page are executed with code segments in the current simulation; 11-5

## **IsPageIncluded,**

Verifies if a CPN page is included in the current simulation; 11-2

## **IsPageObserv,**

Verifies if a CPN page is observable in the current simulation; 11-4

## **IsPagePlaceFus,**

Verifies if a string of characters is used as a CPN place page or instance fusion set name; 10-3

## **IsPagePrime,**

Verifies if a CPN page is a prime page; 11-14

## **IsPageProposed,**

Verifies if a CPN page participates in the occurrence set calculation for the current simulation; 11-3

## **IsPort,**

Verifies if a CPN place or transition is a CPN port node; 10-4

## **IsSubTransAuto,**

Verifies if the subpage of a CPN substitution transition participates in the automatic run for the current simulation; 11-12

## **IsSubTransCode,**

Verifies if the CPN transitions of the subpage of a CPN substitution transition are executed with code segments in the current simulation; 11-11

## **IsSubTransIncluded,**

Verifies if the subpage of a CPN substitution transition is included in the current simulation; 11-8

## **IsSubTransObserv,**

Verifies if the subpage of a CPN substitution transition is observable in the current simulation; 11-10

## **IsSubTransProposed,**

Verifies if the subpage of a CPN substitution transition participates in the occurrence set calculation for the current simulation; 11-9

## L

### **LC\_create,**

Creates a line chart; 15-3

### **LC\_delete,**

Deletes a line chart; 15-8

### **LC\_GetCodeSeg,**

Gets the ID of the CPN code segment region of a line chart; 15-9

### **LC\_init\_chart,**

Initializes a line chart; 15-10

### **LC\_upd\_axisnames,**

Updates the axis names in a line chart.; 15-11

### **LC\_upd\_chart,**

Updates lines in a line chart; 15-12

### **LC\_upd\_line,**

Updates a line of a line chart; 15-13

### **LC\_upd\_linenames,**

Updates the line names in a line chart; 15-14

### **LC\_upd\_pline,**

Updates the fill pattern for a line terminator in a line chart; 15-15

### **LC\_upd\_title,**

Updates the title of a line chart; 15-16

### **LG'create,**

Creates a line graph; A4-2

### **LG'decint,**

Declares an integer line graph; A4-6

### **LG'delete,**

Deletes an integer line graph; A4-7

### **LG'upd\_atag,**

Updates axis labels for an integer line graph; A4-8

### **LG'upd\_line,**

Updates an integer line graph; A4-10

### **LG'upd\_ltag,**

Updates legend labels on an integer line graph; A4-9

### **LG'upd\_pline,**

Updates the pattern of an integer line graph; A4-12

## M

### **MakeArc,**

Converts an existing connector into a CPN arc; 8-8

### **MakeArcExp,**

Converts an auxiliary node into a CPN arc expression region; 8-16

### **MakeCodeSeg,**

Converts an auxiliary node into a CPN code segment region; 8-13

### **MakeColor,**

Converts an auxiliary node into a CPN color set region; 8-10

### **MakeCpnPage,**

Converts a newly created Design page into a CPN page; 8-2

### **MakeGenPort,**

Converts a CPN place or transition into a CPN general port node; 9-5



**MakeGlobalPlaceFus,**

Creates a CPN global fusion set and adds a CPN place to it; 9-8

**MakeGlobDec,**

Converts an auxiliary node into a CPN global declaration; 8-3

**MakeGuard,**

Converts an auxiliary node into a CPN guard region; 8-12

**MakeInitMark,**

Converts an auxiliary node into a CPN initial marking region; 8-11

**MakeInOutPort,**

Converts a CPN place or transition into a CPN input/output port node; 9-4

**MakeInPort,**

Converts a CPN place or transition into a CPN input port node; 9-2

**MakeInstPlaceFus,**

Creates a CPN instance fusion set and adds a CPN place to it; 9-10

**MakeLocDec,**

Converts an auxiliary node into a CPN local declaration; 8-5

**MakeName,**

Converts an auxiliary node into a CPN name region; 8-9

**MakeOutPort,**

Converts a CPN place or transition into a CPN output port node; 9-3

**MakePagePlaceFus,**

Creates a CPN page fusion set and adds a CPN place to it; 9-9

**MakePlace,**

Converts an auxiliary node into a CPN place; 8-6

**MakePrimePage,**

Flags a CPN page node as a prime page; 8-15

**MakeTempDec,**

Converts an auxiliary node into a CPN temporary declaration; 8-4

**MakeTime,**

Converts an auxiliary node into a CPN time region; 8-14

**MakeTrans,**

Converts an auxiliary node into a CPN transition; 8-7

**MakeTransSub,**

Converts a CPN transition into a CPN substitution transition; 9-6

**MC'create,**

Creates a matrix chart; A5-2

**MC'dec,**

Declares an integer matrix chart; A5-5

**MC'delete,**

Deletes an integer matrix chart; A5-6

**MC'fill,**

Updates the fill pattern in an integer matrix chart; A5-7

**MC'upd\_ltag,**

Updates pattern legend labels on an integer matrix chart; A5-9

**MC'write,**

Updates cell values in an integer matrix chart; A5-10

**Miscellaneous Constants,**

Miscellaneous integer constants for use with a variety of functions; 1-4

**N****Node and Connector Shapes,**

Symbolic integer constants for node shapes; 1-5

**Node Types,**

The different types of nodes as integer constants; 1-6

**O****Object Flags,**

Integer flags used to read and set properties of objects; 1-7

**Object Types,**

The different types of objects as integer constants; 1-8

**P****Print Options,**

Constants for use with printing commands; 1-9

**S****SC\_upd\_bar,**

Updates a bar of a snapshot bar chart; 14-18

**SC\_upd\_barnames,**

Updates the barnames in a snapshot bar chart; 14-19

**SC\_upd\_chart,**

Updates bars in a snapshot bar chart; 14-20

**SC\_upd\_part,**

Updates a part for the bars of a snapshot bar chart; 14-21

# Index

---

**SV'avrg,**

Returns the average value of a statistical variable; 12-2

**SV'count,**

Returns the count of the number of times SV'upd has been called on a statistical variable; 12-3

**SV'createint,**

Creates an integer statistical variable; 12-5

**SV'first,**

Returns the first value of a statistical variable; 12-6

**SV'init,**

Initializes a statistical variable; 12-7

**SV'max,**

Returns the max of the values of a statistical variable; 12-8

**SV'min,**

Returns the min of the values of a statistical variable; 12-9

**SV'ss,**

Returns the sum of all squares of the values of a statistical variable; 12-10

**SV'ssd,**

Returns the sum of the squares of deviation of a statistical variable; 12-11

**SV'std,**

Returns the standard deviation of a statistical variable; 12-12

**SV'sum,**

Returns the sum of the values of a statistical variable; 12-13

**SV'upd,**

Updates a statistical variable; 12-14

**SV'value,**

Returns the current value of a statistical variable; 12-15

**SV'vari,**

Returns the variance of a statistical variable; 12-16

## T

**Text Fonts,**

Integer constants for identifying various text fonts; 1-10

**Text Justification,**

Integer constants for identifying the various text justifications; 1-11

**Text Styles,**

Integer constants for identifying the various text styles; 1-12



# **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  
INSCRPTYPE  
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'. |
| dirNODIR      | 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<br>(=STRAIGHTCONN,<br>CURVETOPCONN,<br>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).<br>0 is laser hairline. |
| fill  | Pattern number for interior fill.                                       |
| vis   | Visibility value:<br>FALSE if border is invisible.<br>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<br>(STRAIGHTCONN,<br>CURVETOPCONN,<br>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.<br>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.

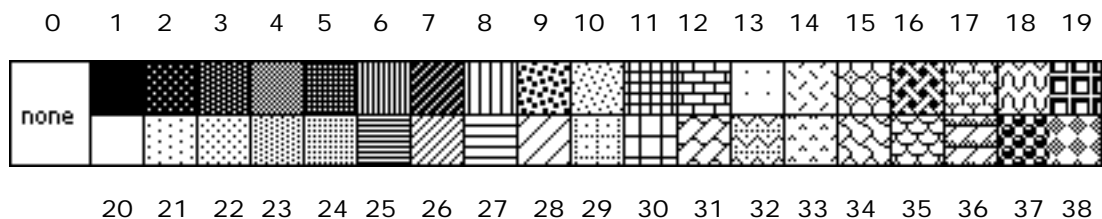
**line** Pattern number for border shading. For Macintosh versions, specify a pattern number for the border shading, using the values shown below for 'fill'.

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.<br>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

|       |   |
|-------|---|
| nodes | ID list of nodes.                           |
| on    | 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.



# **PART 2**

## **Diagram Functions**



# Introduction to Part 2

## Contents of Part 2

Part 2 is divided into the following chapters:

Converting Auxiliary Objects to CPN Objects

Creating CPN Hierarchies

Accessing CPN Diagram Structure

Obtaining CPN Simulation Information

## Feedback

The feedback from Design/CPN separates and surrounds names, numbers, and other indicators with certain characters. Those characters are:

? \* @ : -> ( ) [ ] { }

Do not use these characters in names if you want to receive reasonable looking feedback.





# **Chapter 8**

## **Converting Auxiliary Objects to CPN Objects**

### **Converting Auxiliary Objects to CPN Objects**

The functions described in this chapter convert new pages into CPN model pages, and graphical objects into CP net components.

The dictionary of conversion functions begins on the next page.

### MakeCpnPage

Converts a newly created Design page into a CPN page.

#### Synopsis

```
MakeCpnPage : gid -> unit  
exception cpndbFail : string
```

#### Description

This routine takes an existing page created by means of `DSStr_NewPage` and makes it become a CPN page, so that it will have a page node on the diagram hierarchy page, and it can be used as a subpage for substitution transition.

#### Arguments

ID of Design page

#### Return value

None.

#### Exceptions

Raised if unsuccessful.

### MakeGlobDec

Converts an auxiliary node into a CPN global declaration.

#### Synopsis

```
MakeGlobDec : gid -> unit  
exception cpndbFail : string
```

#### Description

This function takes an existing node created by means of `DSStr_CreateNode` and makes it become a CPN global declaration. If there is text in the node, it will become the text of the CPN global declaration.

#### Arguments

ID of auxiliary node .

#### Return value

None.

#### Exceptions

Raised if unsuccessful.

#### Related Functions

`DSStr_CreateNode`.

### MakeTempDec

Converts an auxiliary node into a CPN temporary declaration.

#### Synopsis

```
MakeTempDec : gid -> unit  
exception cpndbFail : string
```

#### Description

This function takes an existing node created by means of `DSStr_CreateNode` and makes it become a CPN temporary declaration. If there is text in the node, it will become the text of the CPN temporary declaration.

#### Arguments

ID of auxiliary node.

#### Return value

None.

#### Exceptions

Raised if unsuccessful.

#### Related Functions

`DSStr_CreateNode`.

### MakeLocDec

Converts an auxiliary node into a CPN local declaration.

#### Synopsis

```
MakeLocDec : {id : gid, pageid : gid} -> unit  
exception makeLocDec : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing node created by means of `DSStr_CreateNode` and an existing CPN page and makes the node become a CPN local declaration for the CPN page. If there is text in the node, it will become the text of the CPN local declaration. The node must be on the CPN page.

#### Arguments

|                     |                      |
|---------------------|----------------------|
| <code>id</code>     | ID of auxiliary node |
| <code>pageid</code> | ID of CPN page       |

#### Return value

None.

#### Exceptions

|                         |  |
|-------------------------|--|
| <code>makeLocDec</code> | raised if <code>pageid</code> is not a valid CPN page id |
| <code>cpndbFail</code>  | raised if unsuccessful.                                  |

### MakePlace

Converts an auxiliary node into a CPN place.

#### Synopsis

```
MakePlace : gid -> unit  
exception cpndbFail : string
```

#### Description

This function takes an existing node created by means of `DSStr_CreateNode` and makes it become a CPN place. If the place will have no CPN name region, the text in the auxiliary node will be used to generate the place name.

#### Arguments

ID of auxiliary node.

#### Return value

None.

#### Exceptions

Raised if unsuccessful.

### MakeTrans

Converts an auxiliary node into a CPN transition.

#### Synopsis

```
MakeTrans : gid -> unit  
exception cpndbFail : string
```

#### Description

This function takes an existing node created by means of `DSStr_CreateNode` and makes it become a CPN transition. If the transition will have no CPN name region, the text in the auxiliary node will be used to generate the transition name.

#### Arguments

ID of auxiliary node.

#### Return value

None.

#### Exceptions

Raised if unsuccessful.



### MakeArc

Converts an existing connector into a CPN arc.

#### Synopsis

```
MakeArc : gid -> unit  
exception cpndbFail : string
```

#### Description

This routine takes an existing connector created by means of `DSStr_CreateConn` and makes it become a CPN arc.

#### Arguments

ID of auxiliary connector.

#### Return value

None.

#### Exceptions

Raised if unsuccessful.

### MakeName

Converts an auxiliary node into a CPN name region.

### Synopsis

```
MakeName : {cpnnodeid:gid,id:gid} -> unit  
exception makeName : unit  
exception cpndbFail : string
```

### Description

This function takes an existing label created by means of `DSStr_CreateLabel` and makes it become a CPN name region for a CPN place or for a CPN transition. The text of the label will become the text of the CPN name region.

### Arguments

|                        |                                |
|------------------------|--------------------------------|
| <code>cpnnodeid</code> | ID of CPN place or transition. |
| <code>id</code>        | ID of auxiliary node.          |

### Return value

None.

### Exceptions

|                        |   |
|------------------------|---|
| <code>makeName</code>  | raised if <code>cpnnodeid</code> is not a valid CPN place or transition <code>id</code> |
| <code>cpndbFail</code> | raised if unsuccessful.   |

### MakeColor

Converts an auxiliary node into a CPN color set region.

#### Synopsis

```
MakeColor : {id:gid,placeid:gid} -> unit  
exception makeColor : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing label created by means of `DSStr_CreateLabel` and makes it become a CPN color set region for a CPN place. The text of the label will become the text of the CPN color set region.

#### Arguments

|                      |                       |
|----------------------|-----------------------|
| <code>id</code>      | ID of auxiliary node. |
| <code>placeid</code> | ID of CPN place.      |

#### Return value

None.

#### Exceptions

|                        |   |
|------------------------|---|
| <code>makeColor</code> | raised if <code>placeid</code> is not a valid CPN place id. |
| <code>cpndbFail</code> | raised if unsuccessful.                                     |

### MakeInitMark

Converts an auxiliary node into a CPN initial marking region.

#### Synopsis

```
MakeInitMark : {id:gid,placeid:gid} -> unit  
exception makeInitMark : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing label created by means of `DSStr_CreateLabel` and makes it become a CPN initial marking region for a CPN place. The text of the label will become the text of the CPN initial marking region.

#### Arguments

|                      |                       |
|----------------------|-----------------------|
| <code>id</code>      | ID of auxiliary node. |
| <code>placeid</code> | ID of CPN place.      |

#### Return value

None.

#### Exceptions

|                           |   |
|---------------------------|---|
| <code>makeInitMark</code> | raised if <code>placeid</code> is not a valid CPN place id. |
| <code>cpndbFail</code>    | raised if unsuccessful.                                     |

### MakeGuard

Converts an auxiliary node into a CPN guard region.

#### Synopsis

```
MakeGuard : {id:gid,transid:gid} -> unit  
exception makeGuard : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing label created by means of `DSStr_CreateLabel` and makes it become a CPN guard region for a CPN transition. The text of the label will become the text of the CPN guard region.

#### Arguments

|                      |                       |
|----------------------|-----------------------|
| <code>id</code>      | ID of auxiliary node. |
| <code>transid</code> | ID of CPN transition. |

#### Return value

None.

#### Exceptions

|                        |  |
|------------------------|--|
| <code>makeGuard</code> | raised if <code>transid</code> is not a valid CPN transition id. |
| <code>cpndbFail</code> | raised if unsuccessful.  |

### MakeCodeSeg

Converts an auxiliary node into a CPN code segment region.

#### Synopsis

```
MakeCodeSeg : {id:gid,transid:gid} -> unit  
exception makeCodeSeg : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing node created by means of `DSStr_CreateNode` and makes it become a CPN code segment region for a CPN transition. If there is text in the node, it will become the text of the CPN code segment region.

#### Arguments

`id`        ID of auxiliary node.  
`transid`    ID of CPN transition.

#### Return value

None.

### MakeTime

Converts an auxiliary node into a CPN time region.

#### Synopsis

```
MakeTime : {id:gid,transid:gid} -> unit  
exception makeTime : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing label created by means of `DSStr_CreateLabel` and makes it become a CPN time region for a CPN transition. The text of the label will become the text of the CPN time region.

#### Arguments

|                      |                       |
|----------------------|-----------------------|
| <code>id</code>      | ID of auxiliary node. |
| <code>transid</code> | ID of CPN transition. |

#### Return value

None.

### MakePrimePage

Flags a CPN page node as a prime page.

#### Synopsis

```
MakePrimePage : {id:gid,mult:int} -> unit  
exception cpndbFail : string
```

#### Description

This function takes a page and flags it as a prime page.

#### Arguments

|      |                           |
|------|---------------------------|
| id   | ID of page.               |
| mult | multiplicity of the page. |

#### Return value

None.



### MakeArcExp

Converts an auxiliary node into a CPN arc expression region.

#### Synopsis

```
MakeArcExp : {arcid:gid,id:gid} -> unit  
exception makeArcExp : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing label created by means of `DSStr_CreateLabel` and makes it become a CPN arc expression region for a CPN arc. The text of the label will become the text of the CPN arc expression region.

#### Arguments

|                    |                       |
|--------------------|-----------------------|
| <code>arcid</code> | ID of CPN arc.        |
| <code>id</code>    | ID of auxiliary node. |

#### Return value

None.

#### Exceptions

|                         |   |
|-------------------------|---|
| <code>makeArcExp</code> | raised if <code>arcid</code> is not a valid CPN arc id. |
| <code>cpndbFail</code>  | raised if unsuccessful.                                 |

# Chapter 9

## Creating CPN Hierarchies

### Creating CPN Hierarchies

The functions described in this chapter convert ordinary CPN objects into objects that have a particular significance in the hierarchy of the net.

The dictionary of functions for creating CPN hierarchies begins on the next page.

### MakeInPort

Converts a CPN place or transition into a CPN input port node.

#### Synopsis

```
MakeInPort : gid -> unit  
exception MakePort : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing CPN place or transition created by means of `DSStr_CreateNode` and `MakePlace` or `MakeTrans` and makes it become a CPN input port node.

#### Arguments

ID of CPN place or transition

#### Return value

None.

#### Exceptions

|                        |   |
|------------------------|---|
| <code>MakePort</code>  | raised if id is not a valid CPN place or transition id. |
| <code>cpndbFail</code> | raised if unsuccessful.                                 |

#### Related Functions

`MakePlace`, `MakeTrans`, `AssignPort`.

### MakeOutPort

Converts a CPN place or transition into a CPN output port node.

#### Synopsis

```
MakeOutPort : gid -> unit  
exception MakePort : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing CPN place or transition created by means of `DSStr_CreateNode` and `MakePlace` or `MakeTrans` and makes it become a CPN output port node.

#### Arguments

ID of CPN place or transition

#### Return value

None.

#### Exceptions

|                        |   |
|------------------------|---|
| <code>MakePort</code>  | raised if id is not a valid CPN place or transition id. |
| <code>cpndbFail</code> | raised if unsuccessful.                                 |

#### Related Functions

`MakePlace`, `MakeTrans`, `AssignPort`.

### MakeInOutPort

Converts a CPN place or transition into a CPN input/output port node.

#### Synopsis

```
MakeInOutPort : gid -> unit  
exception MakePort : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing CPN place or transition created by means of `DSStr_CreateNode` and `MakePlace` or `MakeTrans` and makes it become a CPN input/output port node.

#### Arguments

ID of CPN place or transition

#### Return value

None.

#### Exceptions

|                        |   |
|------------------------|---|
| <code>MakePort</code>  | raised if id is not a valid CPN place or transition id. |
| <code>cpndbFail</code> | raised if unsuccessful.                                 |

#### Related Functions

`MakePlace`, `MakeTrans`, `AssignPort`.

### MakeGenPort

Converts a CPN place or transition into a CPN general port node.

#### Synopsis

```
MakeGenPort : gid -> unit  
exception MakePort : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing CPN place or transition created by means of `DSStr_CreateNode` and `MakePlace` or `MakeTrans` and makes it become a CPN general port node.

#### Arguments

ID of CPN place or transition

#### Return value

None.

#### Exceptions

|                        |   |
|------------------------|---|
| <code>MakePort</code>  | raised if id is not a valid CPN place or transition id. |
| <code>cpndbFail</code> | raised if unsuccessful.                                 |

#### Related Functions

`MakePlace`, `MakeTrans`, `AssignPort`.

### MakeTransSub

Converts a CPN transition into a CPN substitution transition.

#### Synopsis

```
MakeTransSub : {subpageid:gid,transid:gid} -> unit  
exception makeTransSub : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing CPN transition created by means of `DSStr_CreateNode` and `MakeTrans` and makes it become a CPN substitution transition.

#### Arguments

|                        |  |
|------------------------|--|
| <code>transid</code>   | ID of transition.  |
| <code>subpageid</code> | ID of CPN page to become the subpage of the substitution transition. |

#### Return value

None.

#### Exceptions

|                           |  |
|---------------------------|--|
| <code>makeTransSub</code> | raised if <code>transid</code> is not a valid CPN transition id or if <code>subpageid</code> is not a valid CPN page id. |
| <code>cpndbFail</code>    | raised if unsuccessful.  |

#### Related Functions

`MakePage`, `MakeTrans`.

### AssignPort

Assigns a CPN port node to a CPN socket node for a CPN compound node.

#### Synopsis

```
AssignPort : {id:gid,port:gid,socket:gid} -> unit  
exception PortAssign : unit  
exception cpndbFail : string
```

#### Description

This function takes an existing CPN socket node for a given CPN compound node and assigns a CPN port node to it .

#### Arguments

|        |                          |
|--------|--------------------------|
| id     | ID of CPN compound node. |
| port   | ID of CPN port node      |
| socket | ID of CPN socket node.   |

#### Return value

None.

#### Exceptions

|            |  |
|------------|--|
| PortAssign | raised if id is not a valid CPN compound node id or if port is not a CPN port node id. |
| cpndbFail  | raised if unsuccessful.  |

#### Related Functions

MakeInPort, MakeOutPort , MakeInOutPort , MakeGenPort ,  
MakeTransSub , MakeTransInv , MakePlaceSub .



### MakeGlobalPlaceFus

Creates a CPN global fusion set and adds a CPN place to it.

#### Synopsis

```
MakeGlobalPlaceFus :  
    {name:string,placeid:gid} -> unit  
exception MakeFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters and a CPN place and creates a CPN global fusion set using the string as the name of the fusion set and adds the CPN place to it.

#### Arguments

|         |  |
|---------|--|
| name    | TEXT to become the name of CPN fusion set. |
| placeid | ID of CPN place.                           |

#### Return value

None.

#### Exceptions

|           |  |
|-----------|--|
| MakeFus   | raised if placeid is not a valid CPN place id or if name is already a fusion set name. |
| cpndbFail | raised if unsuccessful.  |

#### Related Functions

MakePlace, IsGlobalPlaceFus, IsPagePlaceFus, IsGlobalTransFus, IsPageTransFus, AddToGlobalPlaceFus.

### MakePagePlaceFus

Creates a CPN page fusion set and adds a CPN place to it.

#### Synopsis

```
MakePagePlaceFus :  
{name:string,pageid:gid,placeid:gid} -> unit  
exception MakeFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters, a CPN page and a CPN place and creates a CPN page fusion set using the string as the name of the fusion set and adds the CPN place to it.

#### Arguments

|         |  |
|---------|--|
| name    | TEXT to become the name of CPN fusion set. |
| pageid  | ID of CPN page.                            |
| placeid | ID of CPN place.                           |

#### Return value

None.

#### Exceptions

|           |  |
|-----------|--|
| MakeFus   | raised if placeid is not a valid CPN place id or if pageid is not a valid CPN page id or if name is already a fusion set name. |
| cpndbFail | raised if unsuccessful.  |

#### Related Functions

MakeCpnPage, MakePlace, IsGlobalPlaceFus, IsPagePlaceFus, IsGlobalTransFus, IsPageTransFus, AddToPagePlaceFus.

### MakeInstPlaceFus

Creates a CPN instance fusion set and adds a CPN place to it.

#### Synopsis

```
MakeInstPlaceFus :  
{name:string,pageid:gid,placeid:gid} -> unit  
exception MakeFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters, a CPN page and a CPN place and creates a CPN instance fusion set using the string as the name of the fusion set and adds the CPN place to it.

#### Arguments

|         |  |
|---------|--|
| name    | TEXT to become the name of CPN fusion set. |
| pageid  | ID of CPN page.                            |
| placeid | ID of CPN place.                           |

#### Return value

None.

#### Exceptions

|           |  |
|-----------|--|
| MakeFus   | raised if placeid is not a valid CPN place id or if pageid is not a valid CPN page id or if name is already a fusion set name. |
| cpndbFail | raised if unsuccessful.  |

#### Related Functions

MakeCpnPage, MakePlace, IsGlobalPlaceFus, IsPagePlaceFus, IsGlobalTransFus, IsPageTransFus, AddToPagePlaceFus.

### AddToGlobalPlaceFus

Adds a CPN place to a CPN global fusion set.

#### Synopsis

```
AddToGlobalPlaceFus :  
    {name:string,placeid:gid} -> unit  
exception AddToFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters as the name of the fusion set and a CPN place and adds the CPN place to the fusion set.

#### Arguments

|         |                              |
|---------|------------------------------|
| name    | TEXT of CPN fusion set name. |
| placeid | ID of CPN place.             |

#### Return value

None.

#### Exceptions

|           |  |
|-----------|--|
| AddToFus  | raised if placeid is not a valid CPN place id or if name is not already a fusion set name. |
| cpndbFail | raised if unsuccessful.  |

#### Related Functions

MakePlace, MakeGlobalPlaceFus, IsGlobalPlaceFus.

### AddToPagePlaceFus

Adds a CPN place to a CPN page or instance fusion set.

#### Synopsis

```
AddToPagePlaceFus :  
{name:string,pageid:gid,placeid:gid} -> unit  
exception AddToFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters as the name of the fusion set, a CPN page and a CPN place and adds the CPN place to the fusion set.

#### Arguments

|         |                              |
|---------|------------------------------|
| name    | TEXT of CPN fusion set name. |
| pageid  | ID of CPN page.              |
| placeid | ID of CPN place.             |

#### Return value

None.

#### Exceptions

|           |  |
|-----------|--|
| AddToFus  | raised if placeid is not a valid CPN place id or if pageid is not a valid CPN page id or if name is not already a fusion set name. |
| cpndbFail | raised if unsuccessful.  |

#### Related Functions

MakeCpnPage, MakePlace, MakePagePlaceFus, MakeInstPlaceFus, IsPagePlaceFus.

### AddToGlobalTransFus

Adds a CPN transition to a CPN global fusion set.

#### Synopsis

```
AddToGlobalTransFus :  
    {name:string,transid:gid} -> unit  
exception AddToFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters as the name of the fusion set and a CPN transition and adds the CPN transition to the fusion set.

#### Arguments

|         |                              |
|---------|------------------------------|
| name    | TEXT of CPN fusion set name. |
| transid | ID of CPN transition .       |

#### Return value

None.

#### Exceptions

|           |   |
|-----------|---|
| AddToFus  | raised if transid is not a valid CPN transition id or if name is not already a fusion set name. |
| cpndbFail | raised if unsuccessful.   |

#### Related Functions

MakeTrans, MakeGlobalTransFus, IsGlobalTransFus.

### AddToPageTransFus

Adds a CPN transition to a CPN page or instance fusion set.

#### Synopsis

```
AddToPageTransFus :  
{name:string,pageid:gid,transid:gid} -> unit  
exception AddToFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters as the name of the fusion set, a CPN page and a CPN transition and adds the CPN transition to the fusion set.

#### Arguments

|         |                                  |
|---------|----------------------------------|
| name    | TEXT of the CPN fusion set name. |
| pageid  | ID of CPN page.                  |
| transid | ID of CPN transition .           |

#### Return value

None.

#### Exceptions

|           |   |
|-----------|---|
| AddToFus  | raised if placeid is not a valid CPN transition id or if pageid is not a valid CPN page id or if name is not already a fusion set name. |
| cpndbFail | raised if unsuccessful.   |

#### Related Functions

MakeCpnPage, MakeTrans, MakePageTransFus,  
MakeInstTransFus, IsPageTransFus.

# **Chapter 10**

## **Accessing CPN Diagram Structure**

### **Accessing CPN Diagram Structure**

The functions described in this chapter return the CPN attributes, as distinct from purely graphical attributes, of objects in a CP net.

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



### IsGlobalPlaceFus

Verifies if a string of characters is used as a CPN place global fusion set name.

#### Synopsis

```
IsGlobalPlaceFus : string -> bool  
exception cpndbFail : string
```

#### Description

This function takes a string of characters and returns true if it is already used as a CPN place global fusion set name.

#### Arguments

TEXT.

#### Return value

TRUE if the string is used as a CPN place global fusion set name.  
FALSE otherwise.

#### Exceptions

cpndbFail      raised if unsuccessful.

#### Related Functions

MakeGlobalPlaceFus.

### IsPagePlaceFus

Verifies if a string of characters is used as a CPN place page or instance fusion set name.

#### Synopsis

```
IsPagePlaceFus : {name:string,pageid:gid} -> bool  
exception isPagePlaceFus : unit  
exception cpndbFail : string
```

#### Description

This function takes a string of characters and a CPN page and returns true if the string is already used as a CPN place page or instance fusion set name.

#### Arguments

|        |                 |
|--------|-----------------|
| name   | TEXT.           |
| pageid | ID of CPN page. |

#### Return value

TRUE if the string is used as a CPN place page or instance fusion set name.

FALSE otherwise.

#### Exceptions

|                |  |
|----------------|--|
| isPagePlaceFus | raised if pageid is not a valid CPN page id. |
| cpndbFail      | raised if unsuccessful.                      |

#### Related Functions

MakePagePlaceFus, MakeInstancePlaceFus.

### IsPort

Verifies if a CPN place or transition is a CPN port node.

#### Synopsis

```
IsPort : gid -> bool  
exception isport : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place or transition and returns true if it is a CPN port node.

#### Arguments

ID of CPN place or transition.

#### Return value

TRUE if the CPN place or transition is a CPN port node.  
FALSE otherwise.

#### Exceptions

|           |   |
|-----------|---|
| isport    | raised if ID is not a valid CPN place or transition id. |
| cpndbFail | raised if unsuccessful.                                 |

#### Related Functions

MakeInPort, MakeOutPort, MakeInOutPort, MakeGenPort.

### GetPageName

Gets the text of the name of a CPN page.

#### Synopsis

```
GetPageName : gid -> string  
exception PageName : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns the text of the name of the CPN page.

#### Arguments

ID of CPN page.

#### Return value

Text containing the name of the CPN page.

#### Exceptions

|           |  |
|-----------|--|
| PageName  | raised if pageid is not a valid CPN page id. |
| cpndbFail | raised if unsuccessful.                      |

#### Related Functions

MakeCpnPage.

### GetNameText

Gets the text of the name region of a CPN place or transition.

#### Synopsis

```
GetNameText : gid -> string  
exception getNameText : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place or transition and returns the text of the name region of the CPN place or transition. If there is no CPN name region associated with the CPN place or transition then the text in the CPN node itself will be returned.

#### Arguments

ID of CPN place or transition.

#### Return value

Text containing the name of the CPN place or transition.

#### Exceptions

|             |  |
|-------------|--|
| getNameText | raised if gid is not a valid place or transition id. |
| cpndbFail   | raised if unsuccessful.                              |

#### Related Functions

MakePlace, MakeTrans.

### GetName

Gets the ID of the CPN name region of a CPN place or transition.

### Synopsis

```
GetName : gid -> gid  
exception getName : unit  
exception cpndbFail : string
```

### Description

This function takes a CPN place or transition and returns the name region of the CPN place or transition. If there is no CPN name region associated with the CPN place or transition then 0 will be returned.

### Arguments

ID of CPN place or transition.

### Return value

ID of the CPN name region, or 0 if it does not exist.

### Exceptions

|           |  |
|-----------|--|
| getName   | raised if gid is not a valid CPN place or transition id. |
| cpndbFail | raised if unsuccessful.                                  |

### Related Functions

MakeName.

### GetColor

Gets the ID of the CPN color set region of a CPN place.

#### Synopsis

```
GetColor : gid -> gid  
exception getColor : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place and returns its color set region. If there is no CPN color set region associated with the CPN place then 0 will be returned.

#### Arguments

ID of CPN place.

#### Return value

ID of the CPN color set region or 0 if it does not exist.

#### Exceptions

|           |  |
|-----------|--|
| getColor  | raised if gid is not a valid CPN place id. |
| cpndbFail | raised if unsuccessful.                    |

#### Related Functions

MakeColor.

### GetInitMark

Gets the ID of the CPN initial marking region of a CPN place.

#### Synopsis

```
GetInitMark : gid -> gid  
exception getInitMark : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place and returns its initial marking region. If there is no CPN initial marking region associated with the CPN place then 0 will be returned.

#### Arguments

ID of CPN place.

#### Return value

ID of the CPN initial marking region or 0 if it does not exist.

#### Exceptions

|                          |  |
|--------------------------|--|
| <code>getInitMark</code> | raised if gid is not a valid CPN place id. |
| <code>cpndbFail</code>   | raised if unsuccessful.                    |

#### Related Functions

`MakeInitMark`.



### GetGuard

Gets the ID of the CPN guard region of a CPN transition.

#### Synopsis

```
GetGuard : gid -> gid  
exception getGuard : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN transition and returns its guard region. If there is no CPN guard region associated with the CPN transition then 0 will be returned.

#### Arguments

ID of CPN transition.

#### Return value

ID of the CPN guard region or 0 if it does not exist.

#### Exceptions

|           |   |
|-----------|---|
| getGuard  | raised if gid is not a valid CPN transition id. |
| cpndbFail | raised if unsuccessful.                         |

#### Related Functions

MakeGuard.

### GetTime

Gets the ID of the CPN time region of a CPN transition.

### Synopsis

```
GetTime : gid -> gid  
exception getTime : unit  
exception cpndbFail : string
```

### Description

This function takes a CPN transition and returns its time region. If there is no CPN time region associated with the CPN transition then 0 will be returned.

### Arguments

ID of CPN transition.

### Return value

ID of the CPN time region or 0 if it does not exist.

### Exceptions

|           |   |
|-----------|---|
| getTime   | raised if gid is not a valid CPN transition id. |
| cpndbFail | raised if unsuccessful.                         |

### Related Functions

MakeTime.

### GetCodeSeg

Gets the ID of the CPN code segment region of a CPN transition or chart node.

#### Synopsis

```
GetCodeSeg : gid -> gid  
exception getCodeSeg : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN transition or chart node and returns its code segment region. If there is no CPN code segment associated with the CPN transition or chart node then 0 will be returned.

#### Arguments

ID of CPN transition or chart node.

#### Return value

ID of the CPN code segment or 0 if it does not exist.

#### Exceptions

|            |   |
|------------|---|
| getCodeSeg | raised if gid is not a valid CPN transition or chart node id. |
| cpndbFail  | raised if unsuccessful.                                       |

#### Related Functions

MakeCodeSeg.

### GetArcExp

Gets the ID of the CPN arc expression region of a CPN arc.

#### Synopsis

```
GetArcExp : gid -> gid  
exception getArcExp : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN arc and returns its arc expression region. If there is no CPN arc expression associated with the CPN arc then 0 will be returned.

#### Arguments

ID of CPN arc.

#### Return value

ID of the CPN arc expression or 0 if it does not exist.

#### Exceptions

|           |  |
|-----------|--|
| getArcExp | raised if gid is not a valid CPN arc id. |
| cpndbFail | raised if unsuccessful.                  |

#### Related Functions

MakeArcExp.

### GetPort

Gets the ID of the CPN port region of a CPN place or transition.

#### Synopsis

```
GetPort : gid -> gid  
exception getPort : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place or transition and returns the port region of the CPN place or transition. If there is no CPN port region associated with the CPN place or transition then 0 will be returned.

#### Arguments

ID of CPN place or transition.

#### Return value

ID of the CPN port region or 0 if it does not exist.

#### Exceptions

|           |  |
|-----------|--|
| getPort   | raised if gid is not a valid CPN place or transition id. |
| cpndbFail | raised if unsuccessful.                                  |

#### Related Functions

MakeInPort, MakeOutPort, MakeInOutPort, MakeGenPort.

### GetFusion

Gets the ID of the CPN fusion set region of a CPN place or transition.

#### Synopsis

```
GetFusion : gid -> gid  
exception getFusion : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place or transition and returns the fusion set region of the CPN place or transition. If there is no CPN fusion set region associated with the CPN place or transition then 0 will be returned.

#### Arguments

ID of CPN place or transition.

#### Return value

ID of the CPN fusion set region or 0 if it does not exist.

#### Exceptions

|           |  |
|-----------|--|
| getFusion | raised if gid is not a valid CPN place or transition id. |
| cpndbFail | raised if unsuccessful.                                  |

#### Related Functions

MakeGlobalPlaceFusion, MakePagePlaceFusion,  
MakeInstPlaceFusion, MakeGlobalTransFusion,  
MakePageTransFusion, MakeInstTransFusion.

### GetCpnInfo

Gets information on CPN objects.

#### Synopsis

```
GetCpnInfo : gid ->  
{objsubtyp:ObjSubType,objtyp:ObjType}  
exception cpndbFail : string
```

#### Description

This function takes any CPN object and returns type and subtype information for it. This function cannot be used to determine if a CPN place or transition belongs to a fusion set and cannot be applied to CPN regions.

#### Arguments

ID of CPN object.

#### Return value

|           |   |
|-----------|---|
| objtyp    | Page   DefBox   Place   Trans   Arc  <br>Barchart   Linechart   Unknown.            |
| objsubtyp | Simple   SubTr   SubPl   Glob   Temp   Loc  <br>In   Out   InOut   Gen   Sub   Inv. |

#### Exceptions

Raised if unsuccessful.

# **Chapter 11**

## **Obtaining CPN Simulation Information**

### **Obtaining CPN Simulation Information**

The functions described in the chapter return information about the status of CPN objects, pages, and subpages relative to simulation.

The dictionary of functions for obtaining CPN simulation information begins on the next page.



### IsPageIncluded

Verifies if a CPN page is included in the current simulation.

#### Synopsis

```
IsPageIncluded : gid -> bool  
exception PageModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns true if it is included in the current simulation run.

#### Arguments

ID of CPN page.

#### Return value

TRUE if the CPN page is included in the current simulation.  
FALSE otherwise.

#### Exceptions

|              |   |
|--------------|---|
| PageModeAttr | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### IsPageProposed

Verifies if a CPN page participates in the occurrence set calculation for the current simulation.

#### Synopsis

```
IsPageProposed : gid -> bool  
exception PageModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns true if it participates in the occurrence set calculation for the current simulation run.

#### Arguments

ID of CPN page.

#### Return value

TRUE if the CPN page participates in the occurrence set calculation for the current simulation.

FALSE otherwise.

#### Exceptions

|              |   |
|--------------|---|
| PageModeAttr | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### IsPageObserv

Verifies if a CPN page is observable in the current simulation.

#### Synopsis

```
IsPageObserv : gid -> bool  
exception PageModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns true if it is observable in the current simulation run.

#### Arguments

ID of CPN page.

#### Return value

TRUE if the CPN page is observable in the current simulation.  
FALSE otherwise.

#### Exceptions

|              |   |
|--------------|---|
| PageModeAttr | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### IsPageCode

Verifies if the CPN transitions of a CPN page are executed with code segments in the current simulation.

#### Synopsis

```
IsPageCode : gid -> bool  
exception PageModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns true if its CPN transitions are executed with code segments in the current simulation run.

#### Arguments

ID of CPN page.

#### Return value

TRUE if the transitions of a CPN page are executed with code segments in the current simulation.

FALSE otherwise.

#### Exceptions

|              |   |
|--------------|---|
| PageModeAttr | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### IsPageAuto

Verifies if a CPN page participates in the automatic run for the current simulation.

#### Synopsis

```
IsPageAuto : gid -> bool  
exception PageModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns true if it participates in the automatic run for the current simulation.

#### Arguments

ID of CPN page.

#### Return value

TRUE if the CPN page participates in the automatic run for the current simulation.

FALSE otherwise.

#### Exceptions

|              |   |
|--------------|---|
| PageModeAttr | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### GetPageModeAttr

Gets the simulation mode attributes for a CPN page.

#### Synopsis

```
GetPageModeAttr : gid ->
{auto : bool , code : bool , included : bool ,
  observ : bool , proposed : bool}
exception PageModeAttr : unit
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns its mode attributes for the current simulation.

Arguments

ID of CPN page.

#### Return value

|          |  |
|----------|--|
| auto     | TRUE if the CPN page participates in the automatic run for the current simulation, FALSE otherwise.                  |
| code     | TRUE if the transitions on the CPN page are executed with code segments for the current simulation, FALSE otherwise. |
| included | TRUE if the CPN page is included in the current simulation, FALSE otherwise.   |
| observ   | TRUE if the CPN page is observable in the current simulation, FALSE otherwise.                                       |
| proposed | TRUE if the CPN page participates in the occurrence set calculation for the current simulation, FALSE otherwise.     |

#### Exceptions

|              |   |
|--------------|---|
| PageModeAttr | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### IsSubTransIncluded

Verifies if the subpage of a CPN substitution transition is included in the current simulation.

#### Synopsis

```
IsSubTransIncluded : gid -> bool  
exception SubTransModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN substitution transition and returns true if its subpage is included in the current simulation run.

#### Arguments

ID of CPN substitution transition.

#### Return value

TRUE if the subpage of the CPN substitution transition is included in the current simulation.

FALSE otherwise.

#### Exceptions

|                  |  |
|------------------|--|
| SubTransModeAttr | raised if gid is not a valid CPN substitution transition id. |
| cpndbFail        | raised if unsuccessful.                                      |

### IsSubTransProposed

Verifies if the subpage of a CPN substitution transition participates in the occurrence set calculation for the current simulation.

#### Synopsis

```
IsSubTransProposed : gid -> bool  
exception SubTransModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN substitution transition and returns true if its subpage participates in the occurrence set calculation for the current simulation run.

#### Arguments

ID of CPN substitution transition.

#### Return value

TRUE if the subpage of the CPN substitution transition participates in the occurrence set calculation for the current simulation.

FALSE otherwise.

#### Exceptions

|                  |  |
|------------------|--|
| SubTransModeAttr | raised if gid is not a valid CPN substitution transition id. |
| cpndbFail        | raised if unsuccessful.                                      |



### IsSubTransObserv

Verifies if the subpage of a CPN substitution transition is observable in the current simulation.

#### Synopsis

```
IsSubTransObserv : gid -> bool  
exception SubTransModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN substitution transition and returns true if its subpage is observable in the current simulation run.

#### Arguments

ID of CPN substitution transition.

#### Return value

TRUE if the subpage of the CPN substitution transition is observable in the current simulation.

FALSE otherwise.

#### Exceptions

|                  |  |
|------------------|--|
| SubTransModeAttr | raised if gid is not a valid CPN substitution transition id. |
| cpndbFail        | raised if unsuccessful.                                      |

### IsSubTransCode

Verifies if the CPN transitions of the subpage of a CPN substitution transition are executed with code segments in the current simulation.

#### Synopsis

```
IsSubTransCode : gid -> bool  
exception SubTransModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN substitution transition and returns true if the CPN transitions of its subpage are executed with code segments in the current simulation run.

#### Arguments

ID of CPN substitution transition.

#### Return value

TRUE if the CPN transitions of the subpage of the CPN substitution transition are executed with code segments in the current simulation.

FALSE otherwise.

#### Exceptions

|                  |  |
|------------------|--|
| SubTransModeAttr | raised if gid is not a valid CPN substitution transition id. |
| cpndbFail        | raised if unsuccessful.                                      |

### IsSubTransAuto

Verifies if the subpage of a CPN substitution transition participates in the automatic run for the current simulation.

#### Synopsis

```
IsSubTransAuto : gid -> bool  
exception SubTransModeAttr : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN substitution transition and returns true if its subpage participates in the automatic run for the current simulation.

#### Arguments

ID of CPN substitution transition.

#### Return value

TRUE if the subpage of the CPN substitution transition participates in the automatic run for the current simulation.

FALSE otherwise.

#### Exceptions

|                  |  |
|------------------|--|
| SubTransModeAttr | raised if gid is not a valid CPN substitution transition id. |
| cpndbFail        | raised if unsuccessful.                                      |

### GetSubTransModeAttr

Gets the simulation mode attributes for the subpage of a CPN substitution transition.

#### Synopsis

```
GetSubTransModeAttr : gid ->
{auto : bool, code : bool, included : bool,
 observ : bool, proposed : bool}
exception SubTransModeAttr : unit
exception cpndbFail : string
```

#### Description

This function takes the subpage of a CPN substitution transition and returns its mode attributes for the current simulation.

#### Arguments

ID of CPN substitution transition.

#### Return value

|          |  |
|----------|--|
| auto     | TRUE if the subpage of the CPN substitution transition participates in the automatic run for the current simulation, FALSE otherwise.                  |
| code     | TRUE if the transitions on the subpage of the CPN substitution transition are executed with code segments for the current simulation, FALSE otherwise. |
| included | TRUE if the subpage of the CPN substitution transition is included in the current simulation, FALSE otherwise.   |
| observ   | TRUE if the subpage of the CPN substitution transition is observable in the current simulation, FALSE otherwise.                                       |
| proposed | TRUE if the subpage of the CPN substitution transition participates in the occurrence set calculation for the current simulation, FALSE otherwise.     |

#### Exceptions

|                  |  |
|------------------|--|
| SubTransModeAttr | raised if gid is not a valid CPN substitution transition id. |
| cpndbFail        | raised if unsuccessful.                                      |

### IsPagePrime

Verifies if a CPN page is a prime page.

#### Synopsis

```
IsPrimePage : gid -> bool  
exception PrimePage: unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns true if it is a prime page.

#### Arguments

ID of CPN page.

#### Return value

TRUE if the CPN page is a prime page, FALSE otherwise.

#### Exceptions

|           |   |
|-----------|---|
| PrimePage | raised if gid is not a valid CPN page id. |
| cpndbFail | raised if unsuccessful.                   |

### GetPageMult

Gets the multiplicity of a CPN page.

#### Synopsis

```
GetPageMult: gid -> int  
exception getPageMult: unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns its multiplicity.

#### Arguments

ID of CPN page.

#### Return value

Multiplicity of CPN page.

#### Exceptions

|             |   |
|-------------|---|
| getPageMult | raised if gid is not a valid CPN page id. |
| cpndbFail   | raised if unsuccessful.                   |

### GetPageInsts

Gets the IDs of CPN page instances.

#### Synopsis

```
GetPageInsts: gid -> dbid list  
exception getPageInsts: unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN page and returns the IDs of its page instances. This function will always return the empty list if invoked within the editor.

#### Arguments

ID of CPN page.

#### Return value

List of page instance IDs of CPN page.

#### Exceptions

|              |   |
|--------------|---|
| getPageInsts | raised if gid is not a valid CPN page id. |
| cpndbFail    | raised if unsuccessful.                   |

### GetPageInstName

Gets the name of a CPN page instance.

#### Synopsis

```
GetPageInstName: dbid -> string  
exception cpndbFail : string
```

#### Description

This function takes a CPN page instance and returns its name. The page instance name describes the path in the page hierarchy by which the page instance is created.

#### Arguments

DBID of CPN page instance.

#### Return value

String containing the name of the CPN page instance.

#### Exceptions

cpndbFail            raised if unsuccessful.



### GetPageInstComp

Gets the ID of a CPN page instance compound node.

#### Synopsis

```
GetPageInstComp: dbid -> id  
exception cpndbFail : string
```

#### Description

This function takes a CPN page instance and returns the ID of its compound node.

#### Arguments

DBID of CPN page.

#### Return value

ID of CPN page instance compound node.

#### Exceptions

cpndbFail            raised if unsuccessful.

### GetMarkingCode

Gets the ML code for accessing the marking of a CPN place on a given CPN page instance.

#### Synopsis

```
GetMarkingCode:  
{placeid : gid, pageinstid : dbid } -> string  
exception getMarkingCode : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place and a CPN page instance and returns the ML code for accessing the marking. This code can then be used as argument for a usestring, as in the following case:

```
usestring  
["val mark = " ^ GetMarkingCode { instid=pgdbid,  
placeid=plid } ];
```

When this expression is evaluated, the value mark contains the string representation of the marking of the place `plid` on the page instance `pgdbid`.

#### Arguments

`placeid` ID of CPN place.  
`pageinstid` DBID of CPN page instance.

#### Return value

String containing the ML code for accessing the marking.

#### Exceptions

`getMarkingCode` raised if `placeid` is not a valid CPN place id.

#### See Also

`GetChangeMarkingCode`.

### GetChangeMarkingCode

Gets the ML code for changing the marking of a CPN place on a given CPN page instance.

#### Synopsis

```
GetChangeMarkingCode:  
{placeid : gid, pageinstid : dbid  
 mark : string} -> string  
exception putMarkingCode : unit  
exception cpndbFail : string
```

#### Description

This function takes a CPN place and a CPN page instance and returns the ML code for changing the marking. This code can then be used as argument for a usestring, as in the following case:

```
usestring  
[ "val mark = " ^ GetMarkingCode { instid=pgdbid,  
  placeid=plid } ];
```

When this expression is evaluated, the place `plid` on the page instance `pgdbid` will have the marking contained in the string representation `mark`.

#### Arguments

`placeid` ID of CPN place.  
`pageinstid` DBID of CPN page instance.  
`mark`.

#### Return value

String containing the ML code for changing the marking.

#### Exceptions

`putMarkingCode` raised if `placeid` is not a valid CPN place id.

#### See Also

`GetMarkingCode`.



# **PART 3**

## **Reporting Functions**



# Introduction to Part 3

## Contents of Part 3

Part 3 is divided into the following chapters:

Statistical Variable Functions

Bar Chart Functions

History Chart Functions

Line Chart Functions

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





# Chapter 12

## Statistical Variable Functions

**Table of Statistical Variable Functions**

| Task   | Function  |
|--|---|
| <b>Creating</b>  | SV'createint  |
| <b>Initializing<br/>and clearing</b>   | SV'init   |
| <b>Updating</b>  | SV'upd  |
| <b>Inspecting</b><br>average<br>count of number of values<br>current value<br>first value<br>maximum<br>minimum<br>standard deviation<br>sum<br>sum of the squares<br>sum of the squares of<br>the deviation<br>variance | SV'avrg<br>SV'count<br>SV'value<br>SV'first<br>SV'max<br>SV'min<br>SV'std<br>SV'sum<br>SV'ss<br>SV'ssd<br>SV'vari |

## Reporting Functions

---

### SV'avrg

Returns the average value of a statistical variable.

#### Synopsis

```
SV'avrg : (int Statvar) -> real
```

#### Description

Returns the average value of a statistical variable.

#### Arguments

Statistical variable created by `SV'createint`.

#### Return Value

Average value of Statvar.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'avrg isv_UseRes; -> 4.666666666666667:real
```

#### See Also

```
SV'createint
SV'init
SV'upd
SV'std
SV'vari
```

#### Corresponding real function

```
SV'avrg : (real Statvar) -> real
```

### SV'count

Returns the count of the number of times `SV'upd` has been called on a statistical variable.

#### Synopsis

```
SV'count : (int Statvar) -> int
```

#### Description

Returns the count of the number of times a statistical variable is updated with `SV'upd`.

#### Arguments

Statistical variable created by `SV'createint`.

#### Return Value

Number of times `Statvar` is updated with `SV'upd`.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'count isv_UseRes; -> 3 : int
```

#### See Also

```
SV'createint
SV'init
SV'upd
SV'sum
```

## Reporting Functions

---

### Corresponding real function

```
SV'count : (real Statvar) -> int
```

### SV'createint

Creates an integer statistical variable.

#### Synopsis

```
SV'createint : unit -> int Statvar
```

#### Description

Creates an integer statistical variable.

#### Arguments

None.

#### Return Value

Integer statistical variable

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();  
> val sv_test = Statvar{ Avrg=ref 0.0, Count=ref 0,  
  First=ref 0, Maxi=ref 0, Mini=ref 0, SSum=ref 0,  
  Sx=ref 0, Ssd=ref 0.0, Std=ref 0.0, Value=ref 0,  
  Vari=ref 0.0 } : int Statvar
```

#### See Also

SV'init  
SV'upd

#### Corresponding real function

```
SV'createreal : unit -> real Statvar
```

# Reporting Functions

---

## SV'first

Returns the first value of a statistical variable.

### Synopsis

```
SV'first : (int Statvar) -> int
```

### Description

Returns the first value of a statistical variable.

### Arguments

Statistical variable created by `SV'createint`.

### Return Value

First value of `Statvar`.

### Exceptions

None.

### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'first isv_UseRes; -> 4 : int
```

### See Also

```
SV'createint
SV'init
SV'upd
SV'value
```

### Corresponding real function

```
SV'first : (real Statvar) -> real
```

### SV'init

Initializes a statistical variable.

### Synopsis

```
SV'init : (int Statvar) -> unit
```

### Description

Initializes a statistical variable. Discards all previous values.

### Arguments

Statistical variable created by `SV'createint`.

### Return Value

None.

### Exceptions

None.

### Example

```
val isv_UseRes = SV'createint ();  
SV'init isv_UseRes; -> () : unit
```

### See Also

SV'createint  
SV'upd  
SV'value

### Corresponding real function

```
SV'init : (real Statvar) -> unit
```

## Reporting Functions

---

### SV'max

Returns the max of the values of a statistical variable.

#### Synopsis

```
SV'max : (int Statvar) -> int
```

#### Description

Returns the max of the values of a statistical variable.

#### Arguments

Statistical variable created by `SV'createint`.

#### Return Value

Max of the values of `Statvar`.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'max isv_UseRes; -> 8 : int
```

#### See Also

```
SV'createint
SV'init
SV'upd
SV'min
```

#### Corresponding real function

```
SV'max : (real Statvar) -> real
```



## SV'min

Returns the min of the values of a statistical variable.

### Synopsis

```
SV'min : (int Statvar) -> int
```

### Description

Returns the min of the values of a statistical variable.

### Arguments

Statistical variable created by `SV'createint`.

### Return Value

Min of the values of `Statvar`.

### Exceptions

None.

### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'min isv_UseRes; -> 2 : int
```

### See Also

```
SV'createint
SV'init
SV'upd
SV'max
```

### Corresponding real function

```
SV'min : (real Statvar) -> real
```

## Reporting Functions

---

### SV'ss

Returns the sum of all squares of the values of a statistical variable.

#### Synopsis

```
SV'ss : (int Statvar) -> int
```

#### Description

Returns the sum of all squares of the values of a statistical variable.

#### Arguments

Statistical variable created by `SV'createint`.

#### Return Value

Sum of all squares of the values of `Statvar`.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'ss isv_UseRes; -> 84 : int
```

#### See Also

```
SV'createint
SV'init
SV'upd
SV'sum
```

#### Corresponding real function

```
SV'ss : (real Statvar) -> real
```

## SV'ssd

Returns the sum of the squares of deviation of a statistical variable.

### Synopsis

```
SV'ssd : (int Statvar) -> real
```

### Description

Returns the sum of the squares of deviation of a statistical variable.

### Arguments

Statistical variable created by `SV'createint`.

### Return Value

Sum of the squares of deviation of `Statvar`.

### Exceptions

None.

### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'ssd isv_UseRes; -> 18.666666666666666 : real
```

### See Also

```
SV'createint
SV'init
SV'upd
SV'ss
SV'std
SV'vari
```

### Corresponding real function

```
SV'ssd : (real Statvar) -> real
```

## Reporting Functions

---

### SV'std

Returns the standard deviation of a statistical variable.

#### Synopsis

```
SV'std : (int Statvar) -> real
```

#### Description

Returns the standard deviation of a statistical variable.

#### Arguments

Statistical variable created by `SV'createint`.

#### Return Value

Standard deviation of `Statvar`.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'std isv_UseRes; -> 3.055050463303893 : real
```

#### See Also

```
SV'createint
SV'init
SV'upd
SV'ss
SV'vari
```

#### Corresponding real function

```
SV'std : (real Statvar) -> real
```

## SV'sum

Returns the sum of the values of a statistical variable.

### Synopsis

```
SV'sum : (int Statvar) -> int
```

### Description

Returns the sum of the values of a statistical variable.

### Arguments

Statistical variable created by `SV'createint`.

### Return Value

Sum of the values of `Statvar`.

### Exceptions

None.

### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'sum isv_UseRes; -> 14 : int
```

### See Also

```
SV'createint
SV'init
SV'upd
SV'count
SV'ss
```

### Corresponding real function

```
SV'sum : (real Statvar) -> real
```

## Reporting Functions

---

### SV'upd

Updates a statistical variable.

#### Synopsis

```
SV'upd : (int Statvar * int) -> unit
```

#### Description

Updates a statistical variable.

#### Arguments

Integer statistical variable and integer constant

#### Return Value

None.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();  
SV'init isv_UseRes;  
SV'upd (isv_UseRes, 4); -> () : unit  
SV'value isv_UseRes; -> 4
```

#### See Also

```
SV'createint  
SV'init
```

#### Corresponding real function

```
SV'upd : (real Statvar * real) -> unit
```

### SV'value

Returns the current value of a statistical variable.

### Synopsis

```
SV'value : (int Statvar) -> int
```

### Description

Returns the current value of a statistical variable.

### Arguments

Statistical variable created by `SV'createint`.

### Return Value

Current value of `Statvar`.

### Exceptions

None.

### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'value isv_UseRes; -> 8 : int
```

### See Also

```
SV'createint
SV'init
SV'upd
SV'first
```

### Corresponding real function

```
SV'value : (real Statvar) -> real
```

## Reporting Functions

---

### SV'vari

Returns the variance of a statistical variable.

#### Synopsis

```
SV'vari : (int Statvar) -> real
```

#### Description

Returns the variance of a statistical variable.

#### Arguments

Statistical variable created by `SV'createint`.

#### Return Value

Variance of Statvar.

#### Exceptions

None.

#### Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'vari isv_UseRes; -> 9.333333333333329:real
```

#### See Also

```
SV'createint
SV'init
SV'upd
SV'ssd
SV'std
```

#### Corresponding real function

```
SV'vari : (real Statvar) -> real
```



# Chapter 13

## Overview of Chart Functions

**Table of Chart Functions**

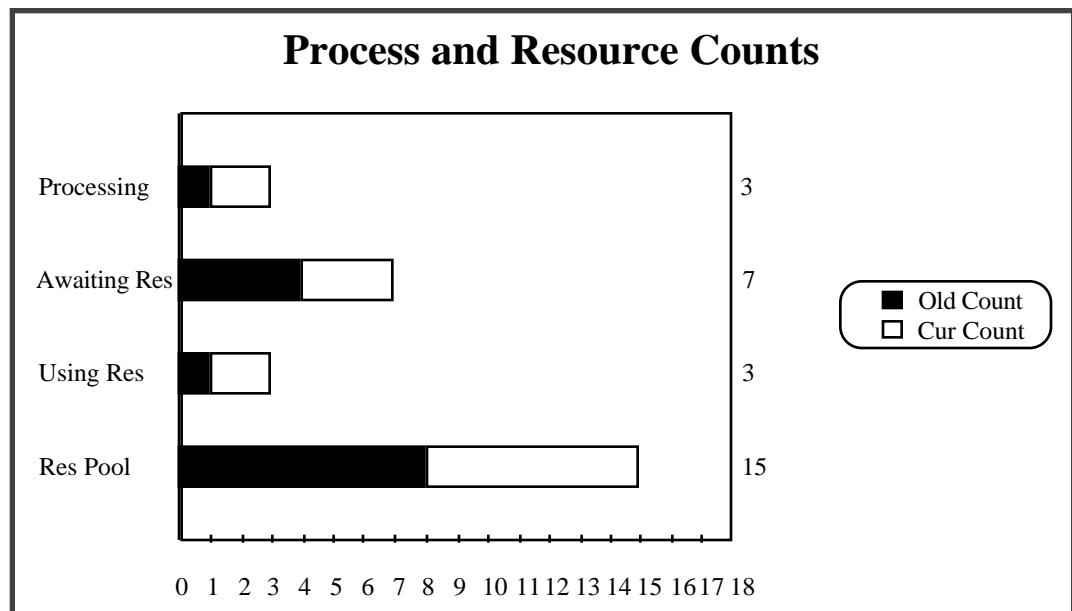
|                     | <b>All<br/>Bar Charts</b>        | <b>History<br/>Bar Charts</b>                 | <b>Snapshot<br/>Bar Charts</b>                               | <b>Line<br/>Charts</b>  |
|---------------------|----------------------------------|---|--|---|
| <b>Creating</b>     | BC_create                        |   |  | LC_create   |
| <b>Updating</b>     | BC_upd_partnames<br>BC_upd_title | HC_upd_bar<br>HC_upd_barnames<br>HC_upd_chart | SC_upd_bar<br>SC_upd_barnames<br>SC_upd_chart<br>SC_upd_part | LC_upd_axisnames<br>LC_upd_chart<br>LC_upd_line<br>LC_upd_linenames<br>LC_upd_pline<br>LC_upd_title |
| <b>Code Segment</b> | BC_GetCodeSeg                    |   |  | LC_GetCodeSeg   |
| <b>Clearing</b>     | BC_clear_chart<br>BC_init_chart  |   |  | LC_init_chart   |
| <b>Deleting</b>     | BC_delete                        |   |  | LC_delete   |



# Chapter 14

## Bar Chart Functions

### Snapshot Chart Example



**Fig. 14-1: Resource Use Model Snapshot Chart ("Res")**

## History Chart Example

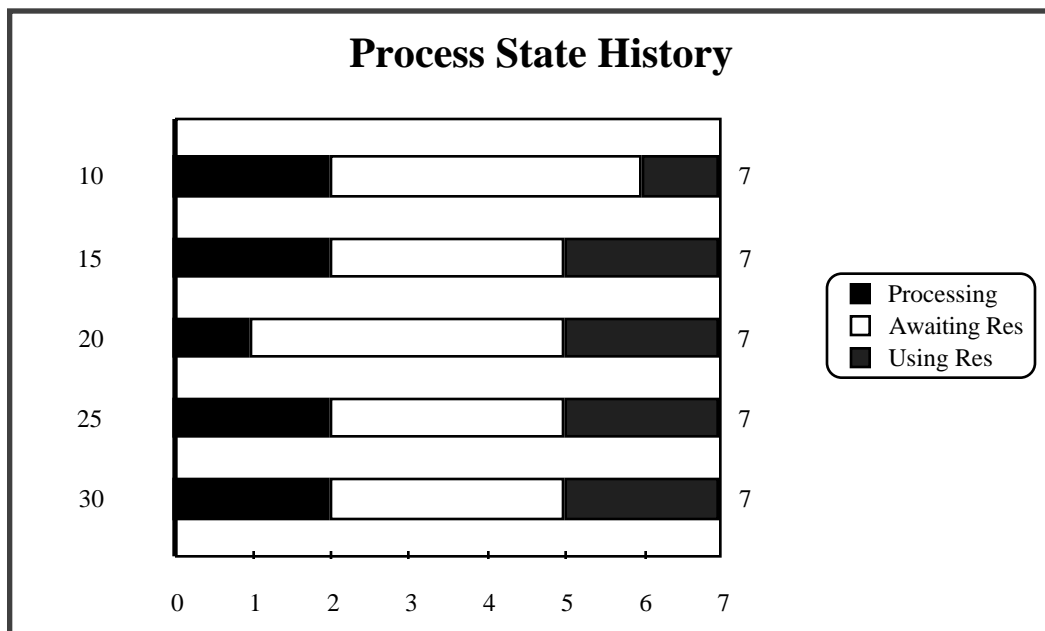


Fig. 14-2: Resource Use Model History Chart (“Hist”)

## Table of Bar Chart Functions

|                     |  |
|---------------------|--|
| <b>Creating</b>     | BC_create  |
| <b>Updating</b>     | BC_upd_partnames<br>BC_upd_title                             |
| <b>History</b>      | HC_upd_bar<br>HC_upd_barnames<br>HC_upd_chart                |
| <b>Snapshot</b>     | SC_upd_bar<br>SC_upd_barnames<br>SC_upd_chart<br>SC_upd_part |
| <b>Code Segment</b> | BC_GetCodeSeg  |
| <b>Clearing</b>     | BC_clear_chart<br>BC_init_chart                              |
| <b>Deleting</b>     | BC_delete  |

## BC\_create

Creates a bar chart.

### Synopsis

```
BC_create : {
  Name:string,      (*Chart identifier, used by all ML
                    functions that reference this chart.
                    The chart name must be unique within
                    the diagram. *)
  Integer:bool,     (*Specifies whether values displayed
                    in a chart are integer (true) or
                    real (false). *)
  NoOfBars:int,     (*Number of bars in the chart. *)
  NoOfParts:int,    (*Number of parts into which each bar
                    is subdivided. *)
  BarHeight:int,    (*Height of the individual bars in
                    pixels. *)
  Hist:bool,        (*Causes the system to generate a
                    history chart. *)
  Retain:int,       (*Number of history chart bars to
                    retain when the bars are moved up
                    during updating. The number must be
                    smaller than or equal to the number
                    specified in NoOfBars and greater
                    than or equal to zero.
  GridNumber:int,   (*Causes the distance between two
                    grid lines to be fixed while the
                    numeric range between the grid lines
                    varies. *)
  GridDist:''a,     (*Causes the numeric range between two
                    grid lines to be fixed while the
                    number of grid lines varies. The
                    range number entered must be of the
                    type specified in the Integer op-
                    tion. *)
  TicksOnly:bool,   (*Specifies whether the grid lines
                    are visible or not. If checked,
                    only small tics can be seen where
                    there are value regions. *)
  Title:bool,       (*Causes the system to generate a
                    title region for the chart. The
                    default title is the string "Title".
                    *)
  Legend:bool,      (*Causes the system to generate a
                    legend describing the bar part pat-
                    terns. *)
  BarNames:bool,    (*Causes the system to generate bar
                    names. *)
  PosValue:bool,    (*Causes the system to generate
                    positive value regions containing
                    default text. The font information
                    may be changed in the Editor. *)
```

## Reporting Functions

---

```
NegValue:bool, (*Causes the system to generate
               negative value regions containing
               default text. The font information
               may be changed in the Editor. *)
UpperValue:bool, (*Causes the system to generate
                 value range regions corresponding to
                 the grid range above the chart. The
                 text in these regions is generated
                 from the Min and Max values and can
                 not be changed by the user. The
                 font can be changed in the Editor.
                 *)
LowerValue:bool, (*Causes the system to generate
                 value range regions corresponding to
                 the grid range below the chart. The
                 text in these regions is generated
                 from the Min and Max values and can
                 not be changed by the user. The
                 font information may be changed in
                 the Editor. *)
SaveCopy:bool, (*Causes the chart to be copied to a
               new page as an Aux node when the
               Initial State (Sim menu) command is
               invoked. *)
KeepCont:bool, (*Specifies whether or not the chart
               will be initialized when the Initial
               State (Sim menu) command is invoked.
               If true, the chart will not be
               initialized when calling Initial
               State. *)
EndOfRun:bool, (*Specifies that the chart is to be
               updated at the end of the simulation
               run. *)
Time:''b,      (*The number of time units between
               chart updates for timed models being
               simulated with time. If both Time
               and Step are included, the chart
               will be updated with both time and
               step intervals. The type is the
               kind of time specified in the
               Simulation Code Options dialog. *)
Step:int,      (*The number of steps between chart
               updates. If both Step and Time are
               included, the chart will be updated
               with both time and step intervals.
               *)
Min:''a,
Max:''a,
               (*Specify the minimum to maximum range
               of values which are initially
               displayed in the plot area. Real
               values must be entered for Min and
               Max if Integer is false. The value
               of Min must not be larger than 0;
               Max must not be smaller than 0.
               *)
Height:int,    (*Height of the chart *)
Width:int,     (*Width of the chart *)
```

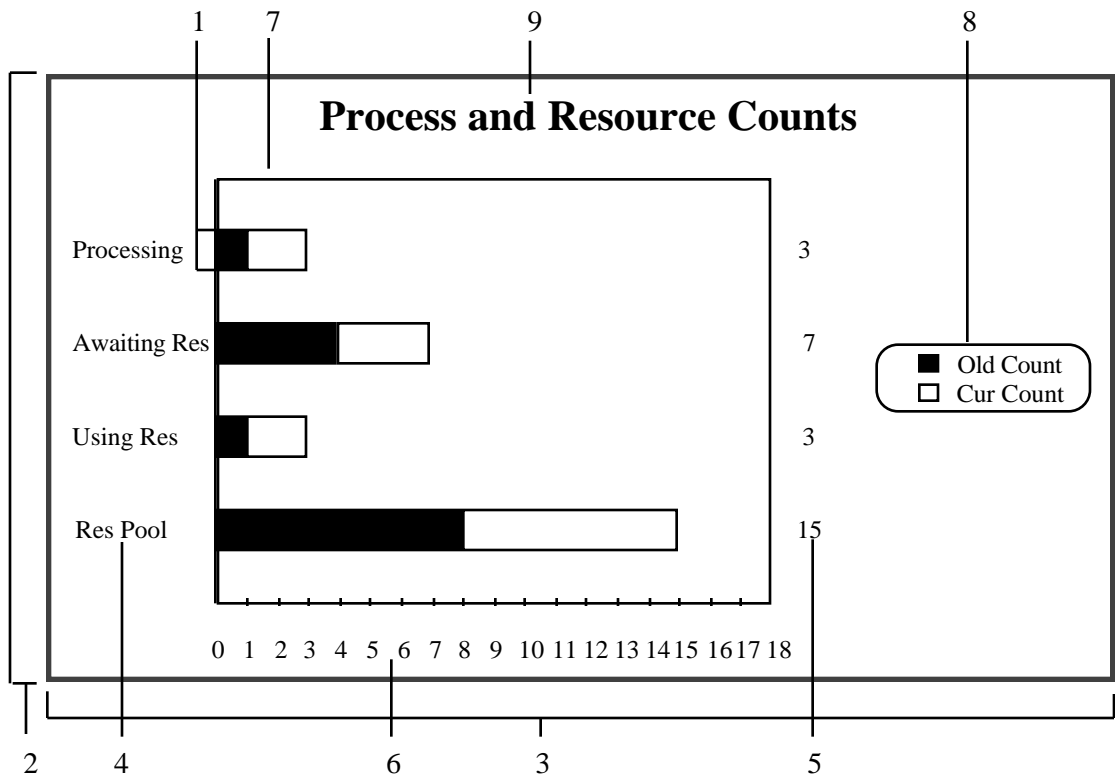
```
x:int,      (*Horizontal location on the page of
              the upper right-hand corner*)
y:int      (*Vertical location on the page of the
              upper right-hand corner*)
} -> string;
```

Description

Creates a bar chart.

Arguments

The Figure Location column in the table below refers to this diagram:



## Reporting Functions

---

| Argument   | Type   | Example | Figure Location | Comments  |
|------------|--------|---------|-----------------|---|
| Name       | string | "Res"   | N/A             | The chart identifier, used by all ML functions that reference this chart.                                     |
| BarHeight  | int    | 20      | 1               | Height of the individual bars in pixels.  |
| Height     | int    | 200     | 2               | Height of the chart in pixels.  |
| Width      | int    | 500     | 3               | Width of the chart in pixels.   |
| x          | int    | 0       | N/A             | Horizontal location on the page of the upper right-hand corner.   |
| y          | int    | 700     | N/A             | Vertical location on the page of the upper right-hand corner.   |
| NoOfParts  | int    | 2       | N/A             | Number of parts in each bar.  |
| NoOfBars   | int    | 4       | N/A             | Number of bars in the chart.  |
| BarNames   | bool   | true    | 4               | Causes the chart to have bar labels.  |
| PosValue   | bool   | true    | 5               | Causes the chart to have positive value labels.   |
| NegValue   | bool   | false   | 5               | Causes the chart to have negative value labels.   |
| Min        | "a     | 0       | N/A             | The minimum value to be displayed in the plot area.   |
| Max        | "a     | 18      | N/A             | The maximum value to be displayed in the plot area.   |
| GridNumber | int    | N/A     | N/A             | Causes the distance between two grid lines to be fixed while the numeric range between the grid lines varies. |
| GridDist   | "a     | 1       | N/A             | Causes the numeric range between two grid lines to be fixed while the number of grid lines varies.            |
| TicksOnly  | bool   | true    | N/A             | Specifies whether or not the grid lines are visible.  |



## Bar Chart Functions

| Argument   | Type         | Example | Figure Location | Comments  |
|------------|--------------|---------|-----------------|---|
| LowerValue | bool         | true    | 6               | Specifies whether or not the chart has grid labels on the bottom of the chart.  |
| UpperValue | bool         | false   | 7               | Specifies whether or not the chart has grid labels at the top of the chart. If the chart has a title, UpperValue is usually false to avoid text conflict.                                   |
| Legend     | bool         | true    | 8               | Specifies whether or not the chart has a legend identifying bar part patterns.  |
| Title      | bool         | true    | 9               | Text of the chart title. The default title can be changed by BC_upd_title.  |
| Hist       | bool         | false   | N/A             | Causes the system to generate a history chart.  |
| Retain     | int          | N/A     | N/A             | The number of history chart bars to retain when the bars are moved up during updating.  |
| SaveCopy   | bool         | false   | N/A             | Causes the chart to be copied to a new page as an Aux node when the Initial State (Sim menu) command is invoked.  |
| KeepCont   | bool         | true    | N/A             | Causes the chart to be initialized when the Initial State (Sim menu) command is invoked.  |
| EndOfRun   | bool         | true    | N/A             | Causes the chart to be updated at the end of the simulation run.  |
| Time       | int/<br>real | 5       | N/A             | The number of time units between chart updates for timed models being simulated with time. If both Time and Step are included, the chart will be updated with both time and step intervals. |
| Step       | int          | N/A     | N/A             | The number of steps between chart updates. If both Step and Time are included, the chart will be updated with both time and step intervals.   |
| Integer    | bool         | true    | N/A             | Specifies whether values displayed in a chart are integer (true) or real (false).   |

### Return Value

Name of the chart.

# Reporting Functions

---

## Exceptions

None.

## Example

To create the integer snapshot bar chart "Res" shown above in Figure 14-1, in integer time mode, type:

```
BC_create
{Name="Res", BarHeight=20, Height=200,
Width=500, x=0, y=700, NoOfParts=2,
NoOfBars=4, BarNames=true, PosValues=true,
NegValues=false, Min=0, Max=18, GridDist=1,
TicksOnly=true, LowerValue=true,
UpperValue=false, Legend=true, Title=true,
Hist=false, SaveCopy=false, KeepCont=true,
EndOfRun=true, Time=5, Integer=true};
BC_upd_title (bc="Res", title="Process and Resource
Counts");
SC_upd_barnames {sc = "Res", tags = ["Processing",
"Awaiting Res", "Using Res", "Res Pool"]};
```

### **BC\_clear\_chart**

Clears a bar chart.

#### **Synopsis**

```
BC_clear_chart : string -> unit
```

#### **Description**

Clears a bar chart.

#### **Arguments**

Name of a bar chart.

#### **Return Value**

None.

#### **Exceptions**

None.

#### **Example**

To clear the bar chart "Res" shown in Figure 14-1, type:

```
BC_clear_chart "Res";
```

### **BC\_delete**

Deletes a bar chart.

#### **Synopsis**

```
BC_delete : string -> unit
```

#### **Description**

Deletes a bar chart.

#### **Arguments**

Name of a bar chart.

#### **Return Value**

None.

#### **Exceptions**

None.

#### **Example**

To delete the bar chart "Res" shown in Figure 14-1, type:

```
BC_delete "Res";
```

### **BC\_GetCodeSeg**

Gets the ID of the CPN code segment region of a bar chart.

#### **Synopsis**

```
BC_GetCodeSeg : string -> gid
```

#### **Description**

Accesses a bar chart code segment region..

#### **Arguments**

Name of a bar chart.

#### **Return Value**

ID of the code segment.

#### **Exceptions**

None.

#### **Example**

To access the code segment of the bar chart "Res" shown in Figure 14-1, type:

```
BC_GetCodeSeg "Res" ;
```

### **BC\_init\_chart**

Initializes a bar chart.

#### **Synopsis**

```
BC_init_chart : string -> unit
```

#### **Description**

Initializes a bar chart.

#### **Arguments**

Name of a bar chart.

#### **Return Value**

None.

#### **Exceptions**

None.

#### **Example**

To initialize the bar chart "Res" shown in Figure 14-1, type:

```
BC_init_chart "Res";
```

### BC\_upd\_partnames

Updates the partnames in a bar chart.

#### Synopsis

```
BC_upd_partnames : { bc:string, tags:string list }  
-> unit
```

#### Description

Establishes the partnames in a bar chart. These names are used in the legend, if any.

#### Arguments

|      |                            |
|------|----------------------------|
| bc   | String name of a bar chart |
| tags | List of string partnames.  |

#### Return Value

None.

#### Exceptions

None.

#### Example

To establish the partnames shown in the legend in Figure 14-1:

```
BC_upd_partnames {bc = "Res", tags = ["Old Count",  
"Cur Count"]};
```

# Reporting Functions

---

## BC\_upd\_title

Updates the title of a bar chart.

### Synopsis

```
BC_upd_title : { bc:string, title:string}
-> unit
```

### Description

Establishes the title of a bar chart.

### Arguments

|       |                               |
|-------|-------------------------------|
| bc    | String name of a bar chart    |
| title | String title for a bar chart. |

### Return Value

None.

### Exceptions

None.

### Example

To establish the title shown in Figure 14-1:

```
BC_upd_title {bc="Res", title= "Process and Resource
Counts"};
```



### HC\_upd\_bar

Updates the current bar of a history bar chart.

#### Synopsis

```
HC_upd_bar:{bc:string, part_values:''a list}  
-> unit
```

#### Description

Updates the current bar of a history bar chart.

#### Arguments

|             |  |
|-------------|--|
| bc          | String name of a bar chart   |
| part_values | List of values for each part of a history chart bar.<br><br>The values must be of the type specified by the Integer option when the chart was created.<br><br>The order is from left to right, and there must be as many elements in the list as there are parts in the bar. |

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the three-part bar of an integer history chart, named "Hist":

```
HC_upd_bar {hc="Hist", part_values=[5,8,2]};
```

## Reporting Functions

---

### HC\_upd\_barnames

Updates the barnames in a history bar chart.

#### Synopsis

```
HC_upd_barnames:{hc:string, tags:string list }  
-> unit
```

#### Description

Updates the barnames in a history bar chart. These names identify the meaning of each bar that appears in the chart.

#### Arguments

```
hc      String name of a history bar chart.  
tags    List of string barnames.
```

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the barnames of a history chart, named "Hist" which has been created with a Retain count of 4:

```
HC_upd_barnames {hc="Hist", tags=["10", "15", "20",  
"25"]};
```

## HC\_upd\_chart

Updates bars in a history bar chart.

### Synopsis

```
HC_upd_chart:{hc:string, values:(''a list * bool)
list, tags:string list }
-> unit
```

### Description

Updates the bars in a history bar chart. More than one bar may be created.

### Arguments

|        |   |
|--------|---|
| hc     | String name of a history bar chart.   |
| values | List of value-specifier tuples, consisting of a (value-list * boolean) pair for each bar. |
| tags   | List of string barnames.  |

### Return Value

None.

### Exceptions

None.

### Example

To update a history chart, named "Hist", which has been created with a Retain count of 4 and whose bars consist of two parts:

```
HC_upd_chart {hc="Hist", values = [([3,4] * true),
([2,7] * true), ([5,6] * true), ([1, 4] * true)],
tags=["10", "15", "20", "25"]};
```

## Reporting Functions

---

### SC\_upd\_bar

Updates a bar of a snapshot bar chart.

#### Synopsis

```
SC_upd_bar:{sc:string, bar:int, part_values:''a list}  
-> unit
```

#### Description

Updates a bar of a history bar chart.

#### Arguments

|             |   |
|-------------|---|
| sc          | String name of a snapshot bar chart.  |
| bar         | Number of the bar to be updated.  |
| part_values | List of values for each part of the snapshot chart bar.<br><br>The values must be of the type specified by the Integer option when the chart was created.<br><br>The order is from left to right, and there must be as many elements in the list as there are parts in the bar. |

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the three parts of the second bar of an integer snapshot chart, named "Res":

```
SC_upd_bar {sc="Res", part_values=[5,8,2]};
```

### SC\_upd\_barnames

Updates the barnames in a snapshot bar chart.

#### Synopsis

```
SC_upd_barnames:{sc:string, tags:string list }  
-> unit
```

#### Description

Updates the barnames in a snapshot bar chart. These names identify the meaning of each bar that appears in the chart.

#### Arguments

```
sc      String name of a snapshot bar chart.  
tags    List of string barnames.
```

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the barnames of a snapshot chart, named "Res" which has four bars:

```
SC_upd_barnames {sc = "Res", tags = ["Processing",  
  "Awaiting Res", "Using Res", "Res Pool"]};
```

# Reporting Functions

---

## SC\_upd\_chart

Updates bars in a snapshot bar chart.

### Synopsis

```
SC_upd_chart:{sc:string, values:(int * 'a list *  
bool) list, tags:string list }  
-> unit
```

### Description

Updates the bars in a snapshot bar chart. Each bar is identified by an integer.

### Arguments

|        |  |
|--------|--|
| sc     | String name of a history bar chart.  |
| values | List of value-specifier tuples, consisting an (int * value-list * boolean) tuple for each bar. |
| tags   | List of string barnames.   |

### Return Value

None.

### Exceptions

None.

### Example

To update a snapshot chart, named "Res", which has 4 bars each consisting of two parts:

```
SC_upd_chart {sc="Res", values = [(1 * [3,4] *  
true), (2 * [2,7] * true), (3 * [5,6] * true), (4 *  
[1, 4] * true)], tags=["10", "15", "20", "25"]};
```

### SC\_upd\_part

Updates a part for the bars of a snapshot bar chart.

#### Synopsis

```
SC_upd_part:{sc:string, part:int, bar_values:''a list}  
-> unit
```

#### Description

Updates a part for the bars of a snapshot bar chart.

#### Arguments

|            |   |
|------------|---|
| sc         | String name of a snapshot bar chart.  |
| part       | Number of the part to be updated.   |
| bar_values | List of values for the specified part of each bar of the snapshot chart.<br><br>The values must be of the type specified by the Integer option when the chart was created.<br><br>There must be as many elements in the list as there are bars. |

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the second part of the four bars of an integer snapshot chart, named "Res":

```
SC_upd_part {sc="Res", part = 3,  
            bar_values=[3,6,2,9]};
```





# Chapter 15

## Line Chart Functions

### Line Chart Example

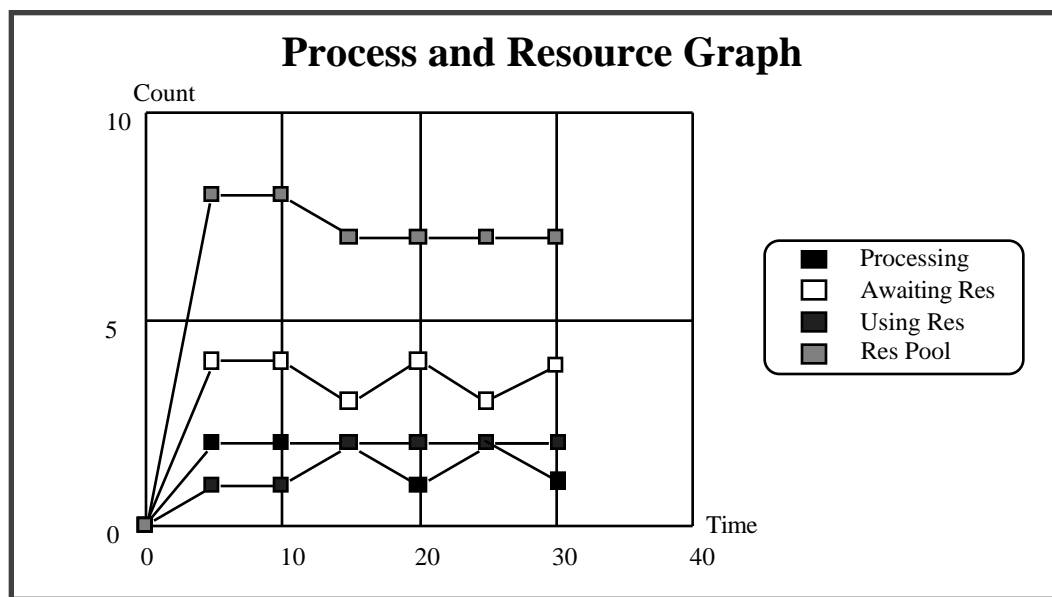


Fig. 15-1: Resource Use Model - Line Chart ("Line")

### Table of Line Chart Functions

|                     |   |
|---------------------|---|
| <b>Creating</b>     | LC_create   |
| <b>Initializing</b> | LC_init_chart   |
| <b>Updating:</b>    | LC_upd_axisnames<br>LC_upd_linenames<br>LC_upd_title<br><br>LC_upd_chart<br>LC_upd_pline<br>LC_upd_line |
| <b>Labels</b>       |   |
| <b>Data</b>         |   |
|                     |   |
|                     |   |

## Reporting Functions

---

|                               |               |
|-------------------------------|---------------|
| <b>Accessing Code Segment</b> | LC_GetCodeSet |
| <b>Deleting</b>               | LC_delete     |

## LC\_create

Creates a line chart.

### Synopsis

```
LC_create: {
  Name:string          (*Chart identifier, which is used by all the
                        ML functions that reference this chart. The
                        chart name must be unique within the
                        diagram. *)
  Integer:bool,        (*Specifies whether values displayed in a
                        chart are integer or real. *)
  NoOfLines:int,       (*Number of chart lines *)
  BoxSize:int,         (*Size of nodes connecting the line chart *)
  StartAtOrigin:bool, (*Lines start at the origin*)
  HorizVert:bool,      (*Specifies how the data points on the graph
                        lines are connected. True causes lines to
                        be drawn with right angles, such as in a
                        time series for hardware circuits. False
                        causes straight lines to be drawn between
                        points. *)
  XNoOfGrids:int,      (*Causes the distance between two grid lines
                        to be fixed while the numeric range between
                        the grid lines varies. *)
  XDist:'a',           (*Causes the numeric range between two grid
                        lines to be fixed while the number of grid
                        lines varies.The number should be of the
                        type specified by Integer. *)
  XTicksOnly:bool,     (*X-axis ticks instead of lines*)
  YNoOfGrids:int,      (*Causes the distance between two grid lines
                        to be fixed while the numeric range between
                        the grid lines varies. *)
  YDist:'a',           (*Causes the numeric range between two grid
                        lines to be fixed while the number of grid
                        lines varies.The number should be of the
                        type specified by Integer. *)
  YTicksOnly:bool,     (*Y-axis ticks instead of lines*)
  Title:string,        (*Text of line chart title*)
  Legend:bool,         (*Specifies whether or not the chart has a
                        legend identifying chart line patterns. *)
  AxisNames:bool,      (*Chart has axis labels. *)
  SaveCopy:bool,       (*Causes the chart to be copied to a new page
                        as an Aux node when the Initial State (Sim
                        menu) command is invoked*)
  KeepCont:bool,       (*Specifies whether or not the chart will be
                        initialized when the Initial State (Sim
                        menu) command is invoked. If true, the
                        chart will not be initialized when calling
                        Initial State. *)
  ReStartAtOrigin:bool,(*Specifies whether or not to start the new
                        lines at the origin or to continue from the
                        end of the previous run's lines*)
  EndOfRun:bool,       (*Specifies that the chart is to be updated at
                        the end of the simulation run. *)
  Time:'b',            (*The number of time units between chart
                        updates for times models being simulated
                        with time. If both Time and Step are
                        specified, the chart will be updated with
                        both time and step intervals. The type is
```

## Reporting Functions

---

```

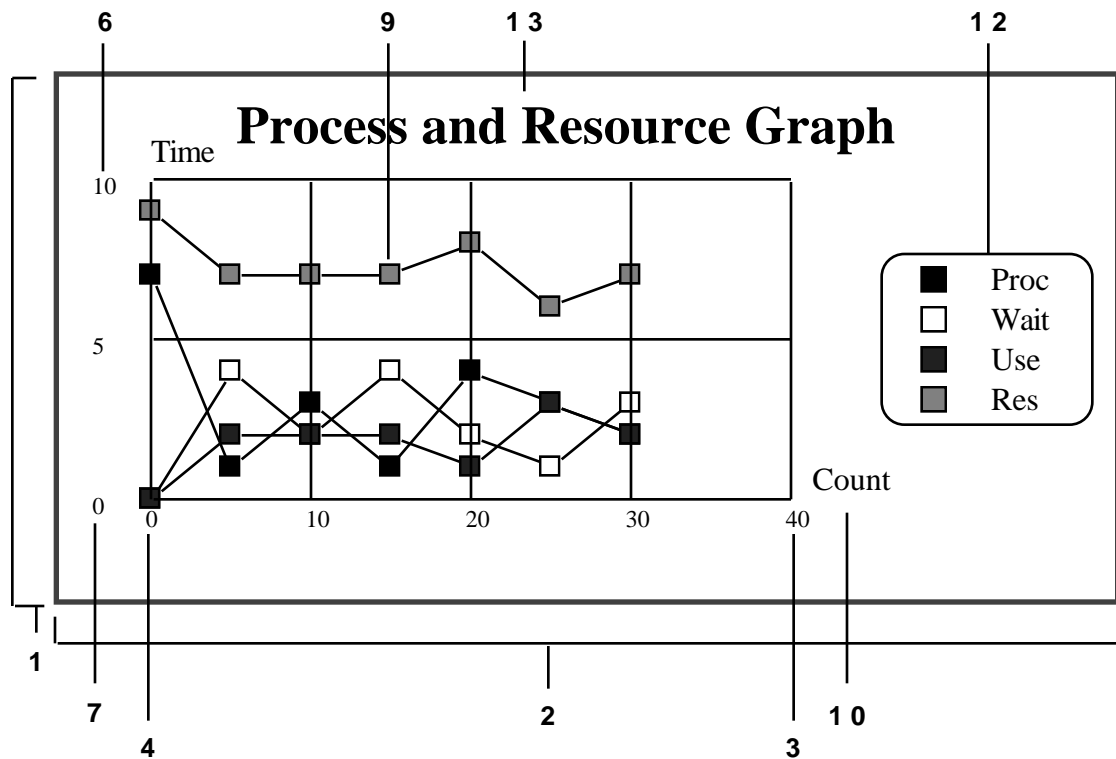
                                the kind of time specified in the Simulation
                                Code Options dialog. *)
Step:int,                      (*The number of steps between chart updates.
                                If both Step and Time are specified, the
                                chart will be updated with both time and
                                step intervals. *)
XMax:''a,                      (*Maximum value of x coordinate*)
XMin:''a,                      (*Minimum value of x coordinate*)
YMax:''a,                      (*Maximum value of y coordinate*)
YMin:''a,                      (*Minimum value of y coordinate*)
XOrigin:''a,
YOrigin:''a,                  (*Specifies where the x and y axes intersect
                                initially. x and y must be located within
                                the XMin, XMax and YMin, YMax range.
                                Specifying MoveAxis can change the location
                                of the origin. *)
RescaleAxis:bool,             (*Causes the system to automatically adjust
                                the chart's display scale to accommodate
                                changes in the data. Rescaling compresses
                                data. *)
MoveAxis:bool                 (*Causes the system to move the display scale
                                to accommodate changes in the data. The
                                display scale is fixed. If a chart will be
                                updated in both the negative and positive
                                direction, this feature should not be used.
                                MoveAxis is comparable to the history chart
                                updating method. *)
Height:int,                   (*Height of the line chart. *)
Width:int,                    (*Width of the line chart. *)
x:int,                        (*Horizontal location on the page of the upper
                                right-hand corner. *)
y:int                         (*Vertical location on the page of the upper
                                right-hand corner. *)
}
-> string
```

### Description

Creates a line chart.

### Arguments

The Figure Location column in the table below refers to this diagram:



**Argument    Type   Ex.   Loc.    Comments**

|               |          |        |     |  |
|---------------|----------|--------|-----|--|
| Name          | string   | "Line" | N/A | Chart identifier, which is used by all the ML functions that reference this chart. The chart name must be unique within the diagram.   |
| NoOfLines     | int      | 4      | N/A | Number of chart lines.   |
| BoxSize       | int      | 20     | 9   | Size of nodes connecting the line chart.   |
| StartAtOrigin | bool     | true   | N/A | Lines start at the origin.   |
| HorizVert     | bool     | false  | N/A | Specifies how the data points on the graph lines are connected. True causes lines to be drawn with right angles, such as in a time series for hardware circuits. False causes straight lines to be drawn between points. |
| XNoOfGrids    | int      | N/A    | N/A | Causes the distance between two grid lines to be fixed while the numeric range between the grid lines varies.  |
| XDist         | int/real | 10     | N/A | Causes the numeric range between two grid lines to be fixed while the number of grid lines varies. The number should be of the type specified by Integer.  |

## Reporting Functions

---

|                 |          |       |     |  |
|-----------------|----------|-------|-----|--|
| XTicksOnly      | bool     | false | N/A | X-axis ticks instead of lines.   |
| YNoOfGrids      | int      | false | N/A | Causes the distance between two grid lines to be fixed while the numeric range between the grid lines varies.  |
| YDist           | "a       | 10    | N/A | Causes the numeric range between two grid lines to be fixed while the number of grid lines varies. The number should be of the type specified by Integer.                                    |
| YTicksOnly      | bool     | false | N/A | Y-axis ticks instead of lines.   |
| Title           | bool     | true  | 13  | Specifies whether or not the chart has a title.  |
| Legend          | bool     | true  | 12  | Specifies whether or not the chart has a legend identifying chart line patterns.   |
| AxisNames       | bool     | true  | 10  | Chart has axis labels.   |
| SaveCopy        | bool     | false | N/A | Causes the chart to be copied to a new page as an Aux node when the Initial State (Sim menu) command is invoked  |
| KeepCont        | bool     | true  | N/A | Specifies whether or not the chart will be initialized when the Initial State (Sim menu) command is invoked. If true, the chart will not be initialized when calling Initial State.          |
| ReStartAtOrigin | bool     | false | N/A | Specifies whether or not to start the new lines at the origin or to continue from the end of the previous run's lines.   |
| EndOfRun        | bool     | true  | N/A | Specifies that the chart is to be updated at the end of the simulation run.  |
| Time            | int/real | 5     | N/A | The number of time units between chart updates for times models being simulated with time. If both Time and Step are specified, the chart will be updated with both time and step intervals. |
| Step            | int      | N/A   | N/A | The number of steps between chart updates. If both Step and Time are specified, the chart will be updated with both time and step intervals.   |
| Integer         | bool     | true  | N/A | Specifies whether values displayed in a chart are integer or real.   |
| XMax            | int/real | 40    | 3   | Maximum value of x coordinate.   |
| XMin            | int/real | 0     | 4   | Minimum value of x coordinate.   |
| YMax            | int/real | 10    | 6   | Maximum value of y coordinate.   |

## Line Chart Functions

|                    |          |        |     |   |
|--------------------|----------|--------|-----|---|
| YMin               | int/real | 0      | 7   | Minimum value of y coordinate.  |
| XOrigin<br>YOrigin | "a<br>"a | 0<br>0 | N/A | Specifies where the x and y axes intersect initially. x and y must be located within the XMin, XMax and YMin, YMax range. Specifying MoveAxis can change the location of the origin.  |
| RescaleAxis        | bool     | true   | N/A | Causes the system to automatically adjust the chart's display scale to accommodate changes in the data. Rescaling compresses data.  |
| MoveAxis           | bool     | false  | N/A | Causes the system to move the display scale to accommodate changes in the data. The display scale is fixed. If a chart will be updated in both the negative and positive direction, this feature should not be used. MoveAxis is comparable to the history chart updating method. |
| Height             | int      | 200    | 1   | Height of the line chart in pixels.   |
| Width              | int      | 500    | 2   | Width of the line chart in pixels.  |
| x                  | int      | 0      | N/A | Horizontal location on the page of the upper right-hand corner.   |
| y                  | int      | 700    | N/A | Vertical location on the page of the upper right-hand corner.   |

### Return Value

Name of chart.

### Exceptions

None.

### Example

To create the integer line chart "Line" shown in Figure 15-1, in integer time mode, type:

```
LC_create
{Name="Line", NoOfLines=4, BoxSize=20,
StartAtOrigin=true, XDist=10, YDist=10, Title=true,
Legend=true, AxisNames=true, KeepCont=true,
EndOfRun=true, Time=5, Integer=true, XMax=40,
XMin=0, YMax=10, YMin=0, XOrigin=0, YOrigin=0,
ReScaleAxis=true, Height=200, Width=500, x=0.
y=700};
```

### **LC\_delete**

Deletes a line chart.

#### **Synopsis**

```
LC_delete : string -> unit
```

#### **Description**

Deletes a line chart.

#### **Arguments**

Reference to a line chart.

#### **Return Value**

None.

#### **Exceptions**

None.

#### **Example**

To delete the line chart "Line" shown in Figure 15-1, type:

```
LC_delete "Line";
```



### LC\_GetCodeSeg

Gets the ID of the CPN code segment region of a line chart.

#### Synopsis

```
LC_GetCodeSeg : string -> gid
```

#### Description

Accesses a line chart code segment region..

#### Arguments

Name of a line chart.

#### Return Value

ID of the code segment.

#### Exceptions

None.

#### Example

To access the code segment of the line chart "Line" shown in Figure 15-1, type:

```
LC_GetCodeSeg "Line";
```

## Reporting Functions

---

### **LC\_init\_chart**

Initializes a line chart.

#### **Synopsis**

```
LC_init_chart : string -> unit
```

#### **Description**

Initializes a line chart.

#### **Arguments**

Name of a line chart.

#### **Return Value**

None.

#### **Exceptions**

None.

#### **Example**

To initialize the line chart "Line" shown in Figure 15-1, type:

```
LC_init_chart "Res";
```

### LC\_upd\_axisnames

Updates the axis names in a line chart.

#### Synopsis

```
LC_upd_axisnames : { lc:string, xtag:string,  
  ytag:string};  
  -> unit
```

#### Description

Establishes the axis names in a line chart. These names are used in the legend, if any.

#### Arguments

```
lc      String name of a line chart.  
xtag    String x-axis name.  
ytag    String y-axis name.
```

#### Return Value

None.

#### Exceptions

None.

#### Example

To establish the axis names shown in the legend in Figure 15-1:

```
LC_upd_axisnames {lc="Line", xtag="Time",  
  ytag="Count"};
```

# Reporting Functions

---

## LC\_upd\_chart

Updates lines in a line chart.

### Synopsis

```
LC_upd_chart:{lc:string, values:(int * 'a * 'a *  
bool * bool) list}  
-> unit
```

### Description

Updates the lines in a line chart.

### Arguments

|        |   |
|--------|---|
| lc     | String name of a line chart.  |
| values | List of value-specifier tuples, consisting an<br>(integer-line-number * x value *<br>y value * true * true) tuple for<br>each line. |

### Return Value

None.

### Exceptions

None.

### Example

To update a line chart, named "Line", which consists of four lines with x,y values: (2,2), (2,3), (2,4), (2,8):

```
LC_upd_chart {lc="Line", values = [(1, 2, 2, true,  
true), (2, 2, 3, true, true), (3, 2, 4, true,  
true), (4, 2, 8, true, true)]};
```

### LC\_upd\_line

Updates a line of a line chart.

#### Synopsis

```
LC_upd_line:{lc:string, line:int, x:''a, y:''a,  
drawline:bool}  
-> unit
```

#### Description

Updates a line of a line chart.

#### Arguments

|          |  |
|----------|--|
| lc       | String name of a line chart.   |
| line     | Number of the line to be updated.  |
| x        | x value for the line.  |
| y        | The value must be of the type specified by the Integer option when the chart was created.<br>y value for the line.                   |
| drawline | The value must be of the type specified by the Integer option when the chart was created.<br>Specifies that the line is to be drawn. |

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the second line of a line chart, named "Line" with x,y value (5,9):

```
LC_upd_line {lc="Line", x=5, y=9,  
drawline=true};
```

## Reporting Functions

---

### LC\_upd\_linenames

Updates the line names in a line chart.

#### Synopsis

```
LC_upd_linenames : { lc:string, tags:string list }  
-> unit
```

#### Description

Establishes the line names in a line chart. These names are used in the legend, if any.

#### Arguments

|      |                              |
|------|------------------------------|
| lc   | String name of a line chart. |
| tags | List of string line names.   |

#### Return Value

None.

#### Exceptions

None.

#### Example

To establish the line names shown in the legend in Figure 15-1:

```
LC_upd_linenames {lc = "Line ", tags =  
  ["Processing", "Awaiting Res", "Using Res", "Res  
  Pool"]};
```

### LC\_upd\_pline

Updates the fill pattern for a line terminator in a line chart.

#### Synopsis

```
LC_upd_pline:{lc:string, line:int, x:''a, y:''a,  
drawline:bool, pattern:int}  
-> unit
```

#### Description

Updates the fill pattern for a line terminator in a line chart.

#### Arguments

|          |  |
|----------|--|
| lc       | String name of a line chart.   |
| line     | Number of the line to be updated.  |
| x        | x value for the line.  |
| y        | The value must be of the type specified by the Integer option when the chart was created.<br>y value for the line. |
| drawline | The value must be of the type specified by the Integer option when the chart was created.                          |
| pattern  | Specifies that the line is to be drawn.<br>The number of the fill pattern.   |

#### Return Value

None.

#### Exceptions

None.

#### Example

To set the fill pattern to 3 for the second line of a line chart, named "Line" with x,y value (5,9):

```
LC_upd_pline {lc="Line", line=2, x=5, y=9,  
drawline=true, pattern=3};
```

# Reporting Functions

---

## LC\_upd\_title

Updates the title of a line chart.

### Synopsis

```
LC_upd_title : { bc:string, title:string}  
-> unit
```

### Description

Establishes the title of a line chart.

### Arguments

|       |                                |
|-------|--------------------------------|
| lc    | String name of a line chart    |
| title | String title for a line chart. |

### Return Value

None.

### Exceptions

None.

### Example

To establish the title shown in Figure 15-1:

```
LC_upd_title {lc="Line", title= "Process and Resource  
Graph"};
```



# **APPENDIX A**

## **Version 1.9 Chart Functions**



# Introduction to Appendix A

## Contents of Appendix A

Appendix A is divided into the following chapters:

Overview of Version 1.9 Chart Functions

Version 1.9 Bar Chart Functions

Version 1.9 History Chart Functions

Version 1.9 Line Chart Functions

Version 1.9 Matrix Chart Functions

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



# Chapter A1

## Overview of Version 1.9 Chart Functions

### Obsolescence of Version 1.9 Chart Functions

In Design/CPN Version 2.0, the charting capabilities provided in Version 1.9 has been replaced by a more sophisticated charting facility. For compatibility, the Version 1.9 charting functions are supported in Version 2.0.

This appendix describes the Version 1.9 charting functions. None of these functions should be used in new models. The functions are documented solely for use in models created with Version 1.9 that already use charts.

Version 1.9 and Version 2.0 charts and chart functions can be used together in the same model. However, 2.0 chart functions cannot be used with 1.9 charts, or vice versa.

### Table of Version 1.9 Chart Functions

|  | <b>Bar<br/>Charts</b>                                  | <b>History<br/>Charts</b>                                   | <b>Line<br/>Charts</b>                                    | <b>Matrix<br/>Charts</b>           |
|--|--|---|---|------------------------------------|
| <b>Creating</b>  | BC'decint<br>BC'create                                 | HC'decint<br>HC'create                                      | LG'decint<br>LG'create                                    | MC'dec<br>MC'create                |
| <b>Updating:<br/>Assigning<br/>labels<br/>Appending<br/>values</b> | BC'upd_ltag<br>BC'upd_rtag<br>BC'upd_col<br>BC'upd_row | HC'hist_ltag<br>HC'hist_rtag<br>HC'hist_init<br>HC'hist_row | LG'upd_ltag<br>LG'upd_atag<br>LG'upd_line<br>LG'upd_pline | MC'upd_ltag<br>MC'fill<br>MC'write |
| <b>Deleting</b>  | BC'delete  | HC'delete   | LG'delete   | MC'delete                          |

### Creating a Chart Page

A newly created chart appears by default on the current page. To cause a chart to be created on a page of its own:

1. Create a page, using `DSStr_NewPage` (described in Chapter 2).
2. Make the page a CPN page, using `MakeCpnPage` (described in Chapter 8).
3. Before creating the chart, make the page current with `DSStr_SetCurPage` (described in Chapter 2).

The chart will appear on the newly created page.

### Positioning a Chart on a Page

The X and Y parameters of the individual chart creation functions specify positioning of the chart on the page. To determine the X and Y coordinates that will position a newly created chart correctly:

1. Create a chart.
2. Move the chart to the desired location on the page. The Reduce menu item of the Page menu is useful in seeing and moving the chart within the page as a whole.
3. Use the Current Object menu item of the Examine menu obtain the X and Y coordinates (`xcenter` and `ycenter`) of the chart.
4. Modify the X and Y parameters of the chart creation function to specify the values obtained in Step 3.

# Chapter A2

## Version 1.9

### Bar Chart Functions

#### Version 1.9 Bar Chart Example

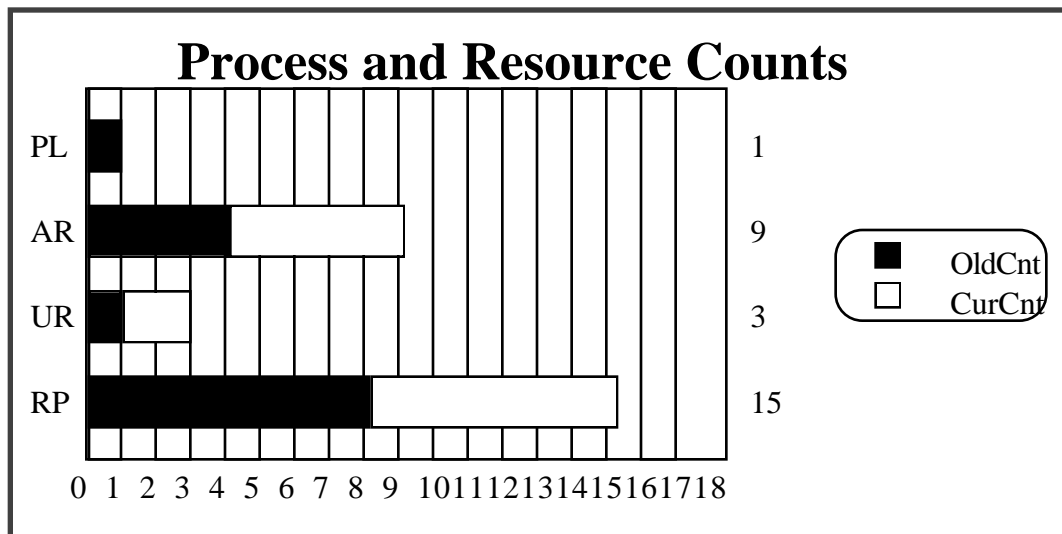


Fig. A2-1: Resource Use Model Bar Chart (bc\_resmod )

#### Table of Version 1.9 Bar Chart Functions

|                         |                            |
|-------------------------|----------------------------|
| <b>Creating</b>         | BC'decint<br>BC'create     |
| <b>Updating:</b>        |                            |
| <b>Assigning labels</b> | BC'upd_ltag<br>BC'upd_rtag |
| <b>Appending values</b> | BC'upd_col<br>BC'upd_row   |
| <b>Deleting</b>         | BC'delete                  |

### BC'create

Creates a bar chart.

#### Synopsis

```
BC'create:
{ bar_height : int,    (* Height of the individual bars (rows) *)
  height : int,        (* Height of the chart *)
  width : int,         (* Width of the chart *)
  x : int,             (* Horizontal location on the page of the
                        upper right-hand corner *)
  y : int,             (* Vertical location on the page of the upper
                        right-hand corner *)
  columns : int,       (* Number of columns in each bar (row) *)
  row : int,           (* Number of bars (rows) in the chart *)
  tag : bool,          (* Specifies whether or not the chart has row
                        labels *)
  val_reg : bool       (* Specifies whether or not the chart has
                        value labels *)
  max_val : int,       (* Determines the range of values that are
                        displayed in the plot area *)
  resize : bool,       (* Specifies whether or not the maximum value
                        shown on the horizontal grid is adjusted to
                        be larger than the maximum bar length *)
  grid_dyn : bool,     (* Specifies whether or not the number of grid
                        lines are to be adjusted dynamically *)
  grid_no : int,       (* Number of grid lines in the chart *)
  grid_vis : bool,     (* Specifies whether or not the grid lines are
                        visible *)
  low_grid : bool,     (* Specifies whether or not the chart has grid
                        labels on the bottom of the chart *)
  up_grid : bool,      (* Specifies whether or not the chart has grid
                        labels at the top of the chart *)
  legend : bool,       (* Specifies whether or not the chart has a
                        legend identifying column patterns *)
  title : string,      (* Text of chart title *)
}
-> int BCHART
```

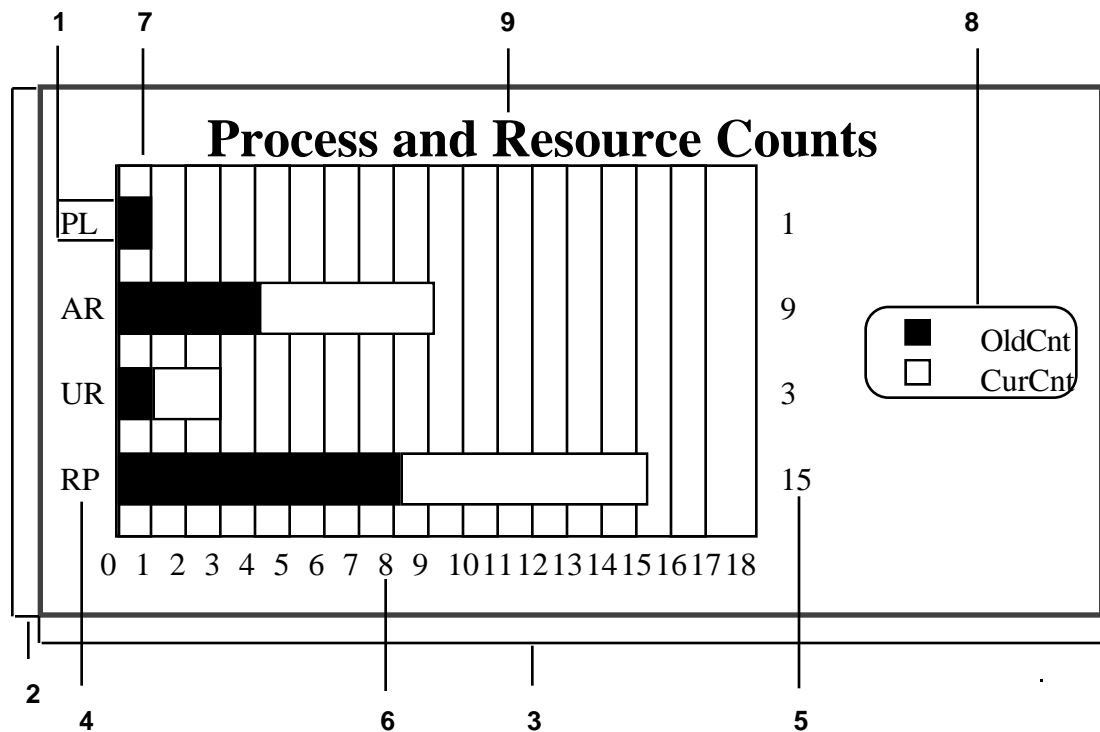
#### Description

Creates a bar chart.

#### Arguments

The Figure Location column in the table below refers to this diagram:





| Argument | Type | Example | Figure Location | Comments |
|----------|------|---------|-----------------|----------|
|----------|------|---------|-----------------|----------|

|            |      |      |     |   |
|------------|------|------|-----|---|
| bar_height | int  | 20   | 1   | Height of the individual bars (rows). The value is in pixels.   |
| height     | int  | 200  | 2   | Height of the chart. The value is in pixels.  |
| width      | int  | 500  | 3   | Width of the chart. The value is in pixels.   |
| x          | int  | 0    | N/A | Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> . |
| y          | int  | 700  | N/A | Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .   |
| columns    | int  | 2    | N/A | Number of columns in each bar (row). The Resource model in this appendix uses two columns in its bar chart.   |
| row        | int  | 4    | N/A | Number of bars (rows) in the chart.   |
| tag        | bool | true | 4   | Specifies whether or not the chart has row labels. BC 'upd_rtag determines the labels for the rows.   |

## Version 1.9 Chart Functions

---

|          |        |            |     |   |
|----------|--------|------------|-----|---|
| val_reg  | bool   | true       | 5   | Specifies whether or not the chart has value labels. The system automatically creates the labels from the position of the left-hand end of the bar.   |
| max_val  | int    | 30         | N/A | Determines the range of values that are displayed in the plot area. Zero is the minimum, max_val is the maximum.  |
| resize   | bool   | true       | N/A | Specifies whether or not the maximum value shown on the horizontal grid is adjusted to be larger than the maximum bar length. Setting resize to true helps to identify errors caused by giving a number outside of the max_val range. |
| grid_dyn | bool   | true       | N/A | Specifies whether or not the number of grid lines are to be adjusted dynamically.   |
| grid_no  | int    | 30         | N/A | Number of grid lines in the chart. If grid_dyn is false, this number is fixed. If grid_dyn is true, this number is the maximum.   |
| grid_vis | bool   | true       | N/A | Specifies whether or not the grid lines are visible.  |
| low_grid | bool   | true       | 6   | Specifies whether or not the chart has grid labels on the bottom of the chart.  |
| up_grid  | bool   | false      | 7   | Specifies whether or not the chart has grid labels at the top of the chart. If the chart has a title, up_grid is usually false to avoid text conflict.  |
| legend   | bool   | true       | 8   | Specifies whether or not the chart has a legend identifying column patterns. BC 'upd_ltag determines the labels for the patterns.   |
| title    | string | See figure | 9   | Text of the chart title.  |

### Return Value

None.

### Exceptions

None.

### Example

To create the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
val bc_resmod = BC'decint ();
bc_resmod := BC'create
{bar_height = 20, height = 200, width =
400, x = 0, y = 350, columns = 2, row = 4,
tag = true, val_reg = true, max_val =
(MaxCount * 2), resize = true, grid_dyn =
true, grid_no = (MaxCount * 2), grid_vis =
true, low_grid = true, up_grid = false,
legend = true, title = "Process and
Resource Counts"};
BC'upd_rtag {bc = bc_resmod, tags = ["PL", "AR",
"UR", "RP"]};
```

### See Also

```
BC'decint
BC'upd_rtag
BC'upd_ltag
BC'delete
```

### Corresponding real function

```
BC'create: { bar_height : int, height : int, width :
int, x : int, y : int, columns : int, tag : bool,
grid_dyn : bool, grid_no : int, grid_vis : bool,
legend : bool, low_grid : bool, max_val : real,
resize : bool, row : int, title : string, up_grid :
bool, val_reg : bool } -> real BCHART
```

### BC'decint

Declares an integer bar chart.

#### Synopsis

```
BC'decint: () -> (int BCHART) ref
```

#### Description

Declares an integer bar chart. This function has the effect of creating an ML value that can be used together with `BC'create` to create an integer bar chart.

#### Arguments

None.

#### Return Value

Returns reference to an integer bar chart.

#### Exceptions

None.

#### Example

To declare the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
val bc_resmod = BC'decint ();
```

#### See Also

`BC'create`

#### Corresponding real function

```
BC'dcreal = fn : () -> (real BCHART) ref
```

### **BC'delete**

Deletes an integer bar chart.

#### **Synopsis**

```
BC'delete : ((int BCHART) ref) -> unit
```

#### **Description**

Deletes an integer bar chart.

#### **Arguments**

Reference to an integer bar chart.

#### **Return Value**

None.

#### **Exceptions**

None.

#### **Example**

To delete the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
BC'delete bc_resmod;
```

#### **See Also**

```
BC'create
```

#### **Corresponding real function**

```
BC'delete : ((real BCHART) ref) -> unit
```

### BC'upd\_col

Updates columns in an integer bar chart.

#### Synopsis

```
BC'upd_col : { bc : (int BCHART) ref, column : int,  
row_values : int list } -> unit
```

#### Description

Updates columns in an integer bar chart.

#### Arguments

|            |                                     |
|------------|-------------------------------------|
| bc         | Reference to an integer bar chart   |
| column     | Integer column number to be updated |
| row_values | List of integer row values.         |

#### Return Value

None.

#### Exceptions

None.

#### Example

The Resource model used in this appendix does not update individual columns, but the function call to update the first column of all the rows would be:

```
BC'upd_col {bc = bc_resmod, column = 1, row_values  
= [(!old_ProcLoc), (!old_AwaitRes), (!old_UseRes),  
(!old_ResPool)]};
```

#### See Also

```
BC'upd_ltag  
BC'upd_row  
BC'upd_rtag
```

### Corresponding real function

```
BC'upd_col : { bc : (real BCHART) ref, column : int,  
row_values : real list } -> unit
```

### BC'upd\_ltag

Updates legend labels on an integer bar chart.

#### Synopsis

```
BC'upd_ltag : { bc : (int BCHART) ref, tags : string  
list } -> unit
```

#### Description

Updates legend labels on an integer bar chart.

#### Arguments

|      |   |
|------|---|
| bc   | Reference to an integer bar chart.      |
| tags | List of strings to use as legend labels |

#### Return Value

None.

#### Exceptions

None.

#### Example

To create legend labels for the bar chart `bc_resmod` shown in Figure A2-1, type:

```
BC'upd_ltag {bc = bc_resmod, tags = ["OldCnt",  
"CurCnt"]};
```

#### See Also

```
BC'upd_col  
BC'upd_row  
BC'upd_rtag
```

#### Corresponding real function

```
BC'upd_ltag : { bc : (real BCHART) ref, tags : string  
list } -> unit
```



### BC'upd\_row

Updates rows in an integer bar chart.

#### Synopsis

```
BC'upd_row : { bc : (int BCHART) ref, row : int,  
column_values : int list } -> unit
```

#### Description

Updates rows in an integer bar chart.

#### Arguments

|               |                                    |
|---------------|------------------------------------|
| bc            | Reference to an integer bar chart. |
| row           | Integer row number to be updated   |
| column_values | List of integer column values.     |

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the second row of the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
old_AwaitRes := SV'value isv_AwaitRes;  
SV'upd (isv_AwaitRes, (!old_AwaitRes) + 1);  
BC'upd_row {bc = bc_resmod,row=2,  
column_values = [(!old_AwaitRes), SV'value  
isv_AwaitRes]};
```

#### See Also

```
BC'upd_col  
BC'upd_ltag  
BC'upd_rtag
```

## Version 1.9 Chart Functions

---

### Corresponding real function

```
BC'upd_row : { bc : (real BCHART) ref, row : int,  
column_values : real list } -> unit
```

### BC'upd\_rtag

Updates row labels on an integer bar chart.

#### Synopsis

```
BC'upd_rtag : { bc : (int BCHART) ref, tags : string  
list } -> unit
```

#### Description

Updates row labels on an integer bar chart.

#### Arguments

|      |                                      |
|------|--------------------------------------|
| bc   | Reference to an integer bar chart.   |
| tags | List of strings to use as row labels |

#### Return Value

None.

#### Exceptions

None.

#### Example

To create row labels for the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
BC'upd_rtag {bc = bc_resmod, tags = ["PL", "AR",  
"UR", "RP"]};
```

#### See Also

```
BC'upd_col  
BC'upd_ltag  
BC'upd_row
```

#### Corresponding real function

```
BC'upd_rtag : { bc : (real BCHART) ref, tags :  
string list } -> unit
```



# Chapter A3

## Version 1.9

### History Chart Functions

#### Version 1.9 History Chart Example

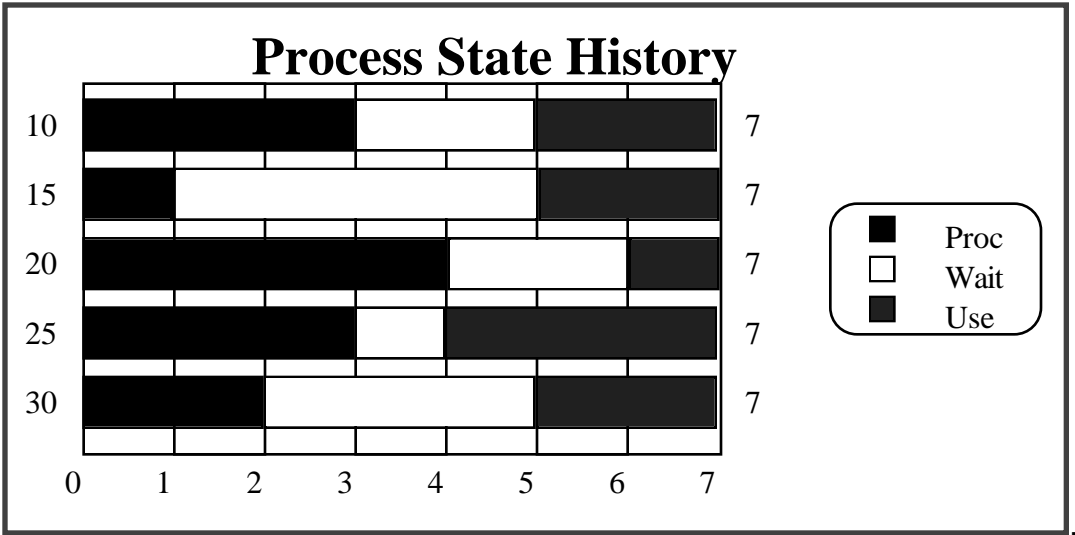


Fig. A3-1: Resource Use Model - History Chart (hc\_resmod )

#### Table of Version 1.9 History Chart Functions

|   |   |
|---|---|
| Creating  | HC'decint<br>HC'create                                      |
| Updating:<br>Assigning labels<br>Appending values | HC'hist_ltag<br>HC'hist_rtag<br>HC'hist_init<br>HC'hist_row |
| Deleting  | HC'delete   |

### HC'create

Creates a history bar chart.

#### Synopsis

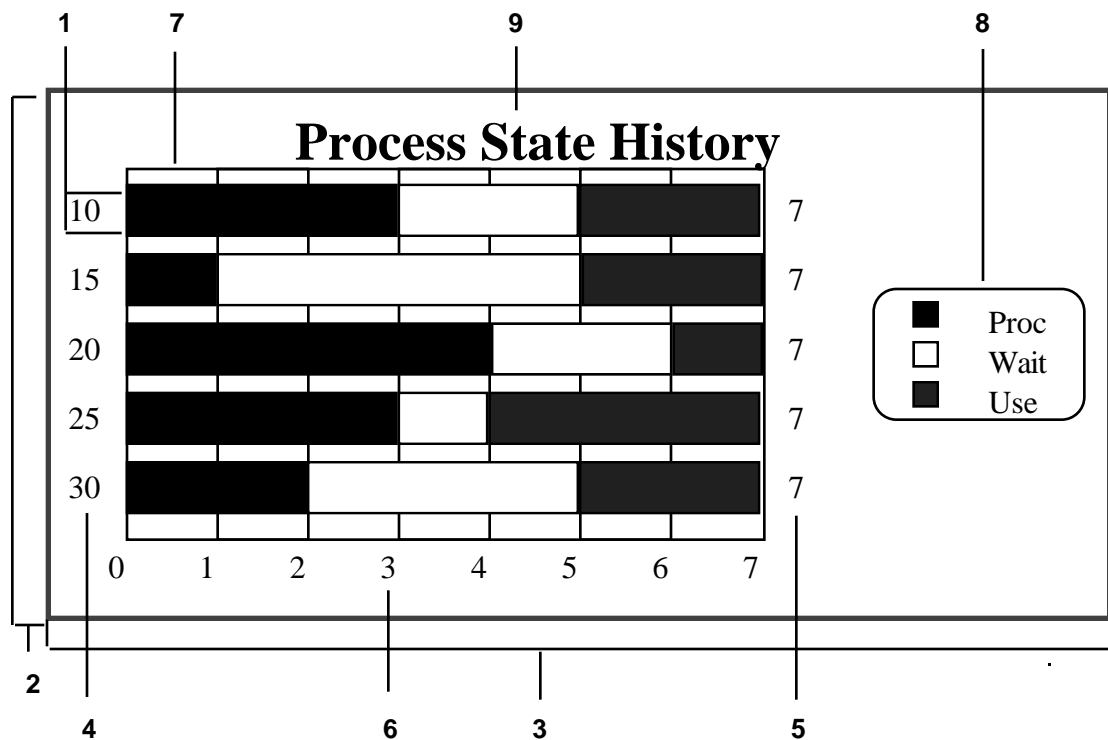
```
HC'create:
{ bar_height : int,    (* Height of the individual rows (bars) *)
  height : int,        (* Height of the chart *)
  width : int,         (* Width of the chart *)
  x : int,             (* Horizontal location on the page of the
                        upper right-hand corner *)
  y : int,             (* Vertical location on the page of the upper
                        right-hand corner *)
  columns : int,       (* Number of columns in each row *)
  row : int,           (* Number of rows in the chart *)
  tag : bool,          (* Specifies whether or not the chart has row
                        labels *)
  val_reg : bool       (* Specifies whether or not the chart has
                        value labels *)
  max_val : int,       (* Determines the range of values that are
                        displayed in the plot area *)
  resize : bool,       (* Specifies whether or not the maximum value
                        shown on the horizontal grid is adjusted to
                        be larger than the maximum bar length *)
  grid_dyn : bool,     (* Specifies whether or not the number of grid
                        lines are to be adjusted dynamically *)
  grid_no : int,       (* Number of grid lines in the chart *)
  grid_vis : bool,     (* Specifies whether or not the grid lines are
                        visible *)
  low_grid : bool,     (* Specifies whether or not the chart has grid
                        labels on the bottom of the chart *)
  up_grid : bool,      (* Specifies whether or not the chart has grid
                        labels at the top of the chart *)
  legend : bool,       (* Specifies whether or not the chart has a
                        legend identifying column patterns *)
  title : string,      (* Text of chart title *)
}
-> int HCHART
```

#### Description

Creates an integer history bar chart.

#### Arguments

The Figure Location column in the table below refers to this diagram:



| Argument   | Type | Example | Figure Location | Comments  |
|------------|------|---------|-----------------|---|
| bar_height | int  | 20      | 1               | Height of the individual rows (bars). The value is in pixels.   |
| height     | int  | 200     | 2               | Height of the chart. The value is in pixels.  |
| width      | int  | 500     | 3               | Width of the chart. The value is in pixels.   |
| x          | int  | 0       | N/A             | Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> . |
| y          | int  | 700     | N/A             | Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .   |
| columns    | int  | 2       | N/A             | Number of columns in each row. The Resource model in this appendix uses three columns in its history chart.   |

## Version 1.9 Chart Functions

|          |        |            |     |  |
|----------|--------|------------|-----|--|
| row      | int    | 4          | N/A | Number of rows in the chart. HC'hist_row generates a new row which is located underneath the previous row. New rows are added up to the maximum specified by this argument. Subsequent rows are added at the bottom and the oldest row is scrolled off the top of the chart. |
| tag      | bool   | true       | 4   | Specifies whether or not the chart has row labels. HC'hist_rtag determines the labels for the rows.  |
| val_reg  | bool   | true       | 5   | Specifies whether or not the chart has value labels. The system automatically creates the labels from the position of the left-hand end of the bar.  |
| max_val  | int    | 30         | N/A | Determines the range of values that are displayed in the plot area. Zero is the minimum, max_val is the maximum.   |
| resize   | bool   | true       | N/A | Specifies whether or not the maximum value shown on the horizontal grid is adjusted to be larger than the maximum bar length. Setting resize to true helps to identify errors caused by giving a number outside of the max_val range.  |
| grid_dyn | bool   | true       | N/A | Specifies whether or not the number of grid lines are to be adjusted dynamically.  |
| grid_no  | int    | 30         | N/A | Number of grid lines in the chart. If grid_dyn is false, this number is fixed. If grid_dyn is true, this number is the maximum.  |
| grid_vis | bool   | true       | N/A | Specifies whether or not the grid lines are visible.   |
| low_grid | bool   | true       | 6   | Specifies whether or not the chart has grid labels on the bottom of the chart.   |
| up_grid  | bool   | false      | 7   | Specifies whether or not the chart has grid labels at the top of the chart. If the chart has a title, up_grid is usually false to avoid text conflict.   |
| legend   | bool   | true       | 8   | Specifies whether or not the chart has a legend identifying column patterns. HC'hist_ltag determines the labels for the patterns.  |
| title    | string | See figure | 9   | Text of the chart title.   |



### Return Value

None.

### Exceptions

None.

### Example

To create the integer history bar chart `hc_resmod` shown in Figure A3-1 type:

```
hc_resmod := HC'create
{bar_height = 20, height = 200, width = 400, x =
0, y = 350, columns = 3, row = 5, tag = true,
val_reg = true, max_val = ProcCount, resize = true,
grid_dyn = false, grid_no = ProcCount, grid_vis =
true, low_grid = true, up_grid = false, legend =
true, title = "Process State History"};
```

### See Also

```
HC'decint
HC'delete
HC'hist_ltag
HC'hist_rtag
```

### Corresponding real function

```
HC'create: { bar_height : int, height : int, width :
int, x : int, y : int, columns : int, tag : bool,
grid_dyn : bool, grid_no : int, grid_vis : bool,
legend : bool, low_grid : bool, max_val : real,
resize : bool, row : int, title : string, up_grid :
bool, val_reg : bool } -> real HCHART
```

### HC'decint

Declares an integer history bar chart.

#### Synopsis

```
HC'decint: () -> (int HCHART) ref
```

#### Description

Declares an integer history bar chart. This function has the effect of creating an ML value that can be used together with `HC'create` to create an integer history chart.

#### Arguments

None.

#### Return Value

Returns reference to integer history bar chart.

#### Exceptions

None.

#### Example

To declare the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
val hc_resmod = HC'decint ();
```

#### See Also

`HC'create`

#### Corresponding real function

```
HC'dcreal = fn : () -> (real HCHART) ref
```

### HC'delete

Deletes an integer history bar chart.

#### Synopsis

```
HC'delete : ((int HCHART) ref) -> unit
```

#### Description

Deletes an integer history bar chart.

#### Arguments

Reference to an integer history chart.

#### Return Value

None.

#### Exceptions

None.

#### Example

To delete the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'delete hc_resmod;
```

#### See Also

```
HC'create
```

#### Corresponding real function

```
HC'delete : ((real HCHART) ref) -> unit
```

### HC'hist\_ltag

Updates column legend labels on an integer history bar chart.

#### Synopsis

```
HC'hist_ltag : { hc : (int HCHART) ref, tags : string  
list } -> unit
```

#### Description

Updates column labels on an integer history bar chart.

#### Arguments

|      |   |
|------|---|
| hc   | Reference to an integer history chart   |
| tags | List of strings to use as column labels |

#### Return Value

None.

#### Exceptions

None.

#### Example

To create legend labels for the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'hist_ltag {hc = hc_resmod , tags = ["Proc",  
"Wait", "Use"]};
```

#### See Also

```
HC'create  
HC'hist_rtag
```

#### Corresponding real function

```
HC'hist_ltag : { hc : (real HCHART) ref, tags : string  
list } -> unit
```

### HC'hist\_init

Initializes columns in an integer history bar chart.

#### Synopsis

```
HC'hist_col : { hc : (int HCHART) ref, init : int } ->  
unit
```

#### Description

Initializes columns in an integer history bar chart. This provides a method for clearing the value of individual columns.

#### Arguments

|      |                                       |
|------|---------------------------------------|
| hc   | Reference to an integer history chart |
| init | Integer column number to be updated   |

#### Return Value

None.

#### Exceptions

None.

#### Example

The Resource Use model used in this appendix does not initialize individual columns, but the function to clear the first column in the current row would be:

```
HC'hist_init {hc = hc_resmod, 1}
```

#### See Also

```
HC'create  
HC'delete  
HC'hist_row
```

## Version 1.9 Chart Functions

---

### Corresponding real function

```
HC'hist_init : { hc : (real HCHART) ref, init : int }  
-> unit
```

### HC'hist\_row

Creates a new row in an integer history bar chart.

#### Synopsis

```
HC'hist_row : { hc : (int HCHART) ref, new_row : int  
list } -> unit
```

#### Description

Creates a new row in an integer history bar chart. New rows are added up to the maximum specified by the HC'create function row argument. Subsequent rows are added at the bottom and the oldest row is scrolled off the top of the chart.

#### Arguments

|         |                                       |
|---------|---------------------------------------|
| hc      | Reference to an integer history chart |
| new_row | Values for the columns in the new row |

#### Return Value

None.

#### Exceptions

None.

#### Example

To create a new row for the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'hist_row {hc = hc_resmod, new_row = [SV'sum  
isv_ProcLoc, SV'sum isv_AwaitRes, SV'sum  
isv_UseRes]};
```

#### See Also

HC'create  
HC'hist\_rtag

## Version 1.9 Chart Functions

---

### Corresponding real function

```
HC'hist_row : { hc : (real HCHART) ref, new_row :  
real list } -> unit
```



### HC'hist\_rtag

Updates row labels on an integer history bar chart.

#### Synopsis

```
HC'hist_rtag : { hc : (int HCHART) ref, tags : string  
list } -> unit
```

#### Description

Updates row labels on an integer history bar chart.

#### Arguments

|      |                                       |
|------|---------------------------------------|
| hc   | Reference to an integer history chart |
| tags | List of strings to use as row labels  |

#### Return Value

None.

#### Exceptions

None.

#### Example

To create time-based row labels for the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'hist_rtag {hc = hc_resmod , tag = makestring  
(time ())};
```

#### See Also

`HC'hist_row`

#### Corresponding real function

```
HC'hist_rtag : { hc : (real HCHART) ref, tags :  
string list } -> unit
```



# Chapter A4

## Version 1.9

### Line Graph Functions

#### Version 1.9 Line Graph Example

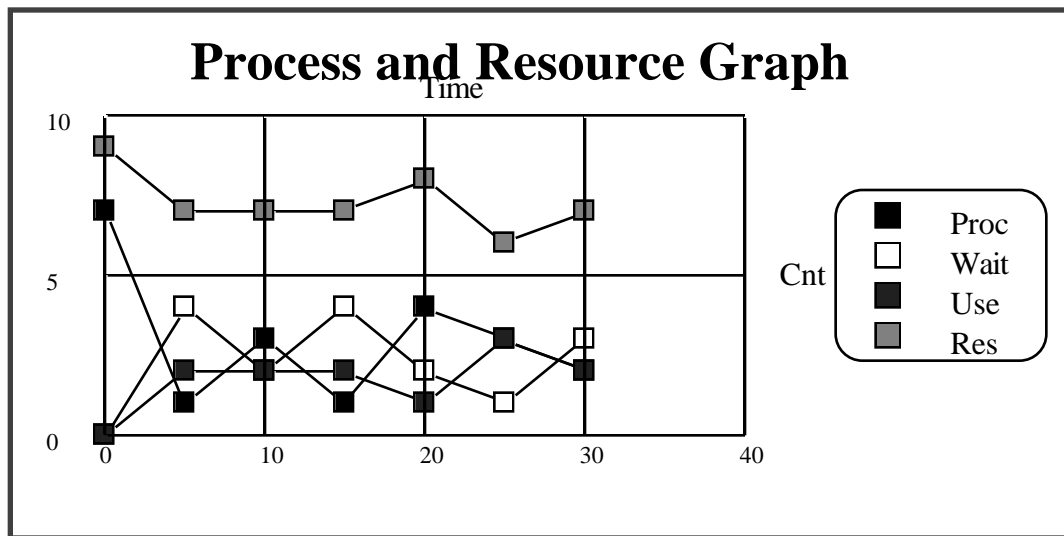


Fig. A4-1: Resource Use Model - Line Graph (lg\_resmod )

#### Table of Version 1.9 Line Graph Functions

|                  |                             |
|------------------|-----------------------------|
| Creating         | LG'decint<br>LG'create      |
| Updating:        |                             |
| Assigning labels | LG'upd_ltag<br>LG'upd_atag  |
| Appending values | LG'upd_line<br>LG'upd_pline |
| Deleting         | LG'delete                   |

### LG'create

Creates a line graph.

#### Synopsis

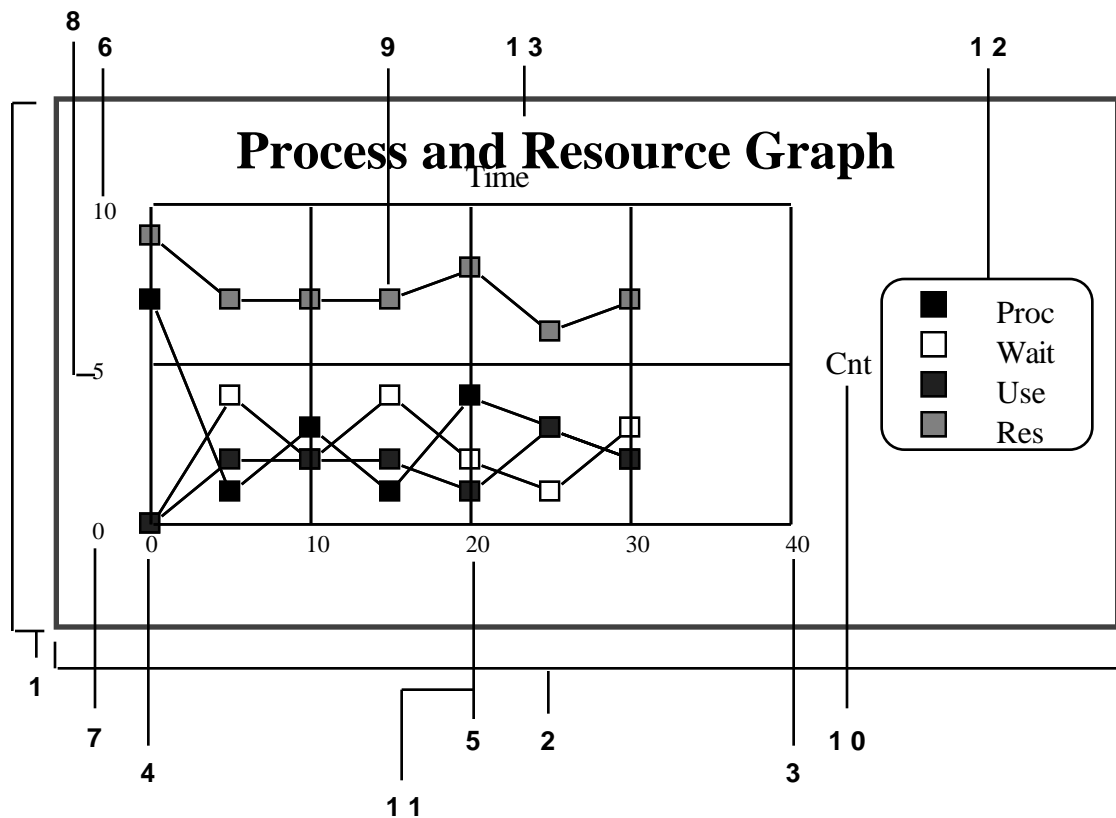
```
LG'create:
{ height : int,      (* Height of the line graph *)
  width : int,       (* Width of the line graph *)
  x : int,           (* Horizontal location on the page of the
                    upper right-hand corner *)
  y : int            (* Vertical location on the page of the upper
                    right-hand corner *)
  xmax : int,        (* Maximum value of x coordinate *)
  xmin : int,        (* Minimum value of x coordinate *)
  xcent : int,       (* x value of center *)
  XMaxTick : int,    (* Number of ticks on the x coord right of
                    center *)
  XMinTick : int,    (* Number of ticks on the x coord left of
                    center *)
  ymax : int,        (* Maximum value of y coordinate *)
  ymin : int,        (* Minimum value of y coordinate *)
  ycent : int,       (* y value of center *)
  YMaxTick : int,    (* Number of ticks of the y coord right of
                    center *)
  YMinTick : int,    (* Number of ticks on the y coord left of
                    center *)
  bigticks : bool,   (* Specifies whether or not ticks are as big
                    as the line graph *)
  multiple : bool,   (* Specifies whether or not the graph is a
                    multiple y coordinate system *)
  nline : int,       (* Number of graph lines *)
  thick : int,       (* Thickness of graph lines *)
  boxsize : int,     (* Size of nodes connecting the line graph *)
  timing : bool,     (* Specifies whether or not the graph is a
                    time-series graph *)
  axis : bool,       (* Specifies whether or not the graph has axis
                    labels *)
  center : bool,     (* Specifies whether or not the graph has
                    center labels *)
  legend : bool,     (* Specifies whether or not the graph has a
                    legend identifying graph line patterns *)
  title : string,    (* Text of line graph title*)
}
-> int LINEGRAPH
```

#### Description

Creates an integer line graph.

#### Arguments

The Figure Location column in the table below refers to this diagram:



| Argument | Type | Example | Figure Location | Comments   |
|----------|------|---------|-----------------|--|
| height   | int  | 200     | 1               | Height of the line graph. The value is in pixels.  |
| width    | int  | 500     | 2               | Width of the line graph. The value is in pixels.   |
| x        | int  | 0       | N/A             | Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses x and y. |
| y        | int  | 700     | N/A             | Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses x and y.   |
| xmax     | int  |         | 3               | Maximum value of x coordinate.   |
| xmin     | int  |         | 4               | Minimum value of x coordinate.   |
| xcent    | int  |         | 5               | x value of center  |
| XMaxTick | int  |         |                 | Number of ticks on the x coordinate right of center  |
| XMinTick | int  |         |                 | Number of ticks on the x coordinate left of center   |

## Version 1.9 Chart Functions

---

|                       |        |            |    |   |
|-----------------------|--------|------------|----|---|
| y <sub>max</sub>      | int    |            | 6  | Maximum value of y coordinate   |
| y <sub>min</sub>      | int    |            | 7  | Minimum value of y coordinate   |
| y <sub>cent</sub>     | int    |            | 8  | y value of center   |
| Y <sub>Max</sub> Tick | int    |            |    | Number of ticks on the y coordinate above the center  |
| Y <sub>Min</sub> Tick | int    |            |    | Number of ticks on the y coordinate below the center  |
| bigticks              | bool   |            |    | Specifies whether or not ticks are as big as the line graph   |
| multiple              | bool   |            |    | Specifies whether or not the graph is a multiple y coordinate system  |
| n <sub>line</sub>     | int    |            |    | Number of graph lines   |
| thick                 | int    |            |    | Thickness of graph lines  |
| boxsize               | int    |            | 9  | Size of nodes connecting the line graph. The boxes contain the patterns identifying the lines.                                      |
| timing                | bool   |            |    | Specifies whether or not the graph is a time-series graph   |
| axis                  | bool   |            | 10 | Specifies whether or not the graph has axis labels  |
| center                | bool   |            | 11 | Specifies whether or not the graph has center labels.   |
| legend                | bool   |            | 12 | Specifies whether or not the graph has a legend describing graph line patterns. LG'upd_ltag determines the labels for the patterns. |
| title                 | string | See figure | 13 | Text of the graph title.  |

### Return Value

None.

### Exceptions

None.

### Example

To create the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
lg_resmod := LG'create
{height=200, width=400, x=0, y=350, xmax=40,
xmin=0, xcent=20, XMaxTick=2, XMinTick=2, ymax=10,
ymin=0, ycent=5, YMaxTick=1, YMinTick=1,
bigticks=true, multiple=false, nline=4, thick=2,
boxsize=10, timing=false, axis=true, center=true,
legend=true, title="Process and Resource Graph"};
```

### See Also

```
LG'decint
LG'delete
LG'upd_atag
LG'upd_ltag
```

### Corresponding real function

```
LG'create: {legend : bool, XMaxTick : int, XMinTick :
int, XMaxTick : int, XMinTick : int, axis : bool,
bigticks : bool, boxsize : int, center : bool, height
: int, multiple : bool, nline : int, thick : int,
timing : bool, title : string, width : int, x : int,
xcent : real, xmax : real, xmin : real, y : int,
ycent : real, ymax : real, ymin : real } -> int
LINEGRAPH
```

### LG'decint

Declares an integer line graph.

#### Synopsis

```
LG'decint: () -> (int LINEGRAPH) ref
```

#### Description

Declares an integer line graph. This function has the effect of creating an ML value that can be used together with LG'create to create an integer line graph.

#### Arguments

None.

#### Return Value

Reference to an integer line graph

#### Exceptions

None.

#### Example

To declare the integer line graph lg\_resmod shown in Figure A4-1, type:

```
val lg_resmod = LG'decint ();
```

#### See Also

LG'create

#### Corresponding real function

```
LG'decreal: () -> (real LINEGRAPH) ref
```



### LG'delete

Deletes an integer line graph.

#### Synopsis

```
LG'delete : ((int LINEGRAPH) ref) -> unit
```

#### Description

Deletes an integer line graph.

#### Arguments

Reference to an integer line graph

#### Return Value

None.

#### Exceptions

None.

#### Example

To delete the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'delete lg_resmod;
```

#### See Also

```
LG'create
```

#### Corresponding real function

```
LG'delete : ((real LINEGRAPH) ref) -> unit
```

### LG'upd\_atag

Updates axis labels for an integer line graph.

#### Synopsis

```
LG'upd_atag: { lg : (int LINEGRAPH) ref, x : string, y  
: string } -> unit
```

#### Description

Updates axis labels for an integer line graph.

#### Arguments

lg Reference to an integer line graph  
x String for x-coordinate tag  
y String for y-coordinate tag

#### Return Value

None.

#### Exceptions

None.

#### Example

To create axis labels for the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'upd_atag {lg = lg_resmod, x="Cnt", y="Time" };
```

#### See Also

LG'create  
LG'upd\_ltag

#### Corresponding real function

```
LG'upd_atag: { lg : (real LINEGRAPH) ref, x : string,  
y : string } -> unit
```

### LG'upd\_ltag

Updates legend labels on an integer line graph.

#### Synopsis

```
LG'upd_ltag : { lg : (int LINEGRAPH) ref, tags :  
string list } -> unit
```

#### Description

Updates labels on an integer line graph.

#### Arguments

|      |   |
|------|---|
| lg   | Reference to an integer line graph      |
| tags | List of strings for lines on line graph |

#### Return Value

None.

#### Exceptions

None.

#### Example

To create legend labels for the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'upd_ltag {lg = lg_resmod, tags =  
["Proc","Wait","Use", "Res"]};
```

#### See Also

```
LG'create  
LG'upd_atag
```

#### Corresponding real function

```
LG'upd_ltag : { lg : (real LINEGRAPH) ref, tags :  
string list } -> unit
```

### LG'upd\_line

Updates an integer line graph.

#### Synopsis

```
LG'upd_line : { lg : (int LINEGRAPH) ref, line : int,  
x : int, y : int} -> unit
```

#### Description

Updates an integer line graph.

#### Arguments

|      |  |
|------|--|
| lg   | Reference to an integer line graph     |
| line | Number of the line to be updated       |
| x    | x-coordinate of the line to be updated |
| y    | y-coordinate of the line to be updated |

#### Return Value

None.

#### Exceptions

None.

#### Example

To update the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'upd_line {lg = lg_resmod, line = 1,  
  x = time (), y = SV'sum isv_ProcLoc};  
LG'upd_line {lg = lg_resmod, line = 2,  
  x = time (), y = SV'sum isv_AwaitRes};  
LG'upd_line {lg = lg_resmod, line = 3,  
  x = time (), y = SV'sum isv_UseRes};  
LG'upd_line {lg = lg_resmod, line = 4,  
  x = time (), y = SV'sum isv_ResPool};
```

### See Also

LG'create  
LG'upd\_pline

### Corresponding real function

LG'upd\_line : { lg : (**real** LINEGRAPH) ref, line : int, x : **real**, y : **real** } -> unit

### LG'upd\_pline

Updates the pattern of an integer line graph.

#### Synopsis

```
LG'upd_pline : { lg : (int LINEGRAPH) ref, line : int,  
pattern : int, x : int, y : int} -> unit
```

#### Description

Updates the pattern of an integer line graph.

#### Arguments

|         |   |
|---------|---|
| lg      | Reference to an integer line graph        |
| line    | Number of the line to be updated          |
| pattern | Pattern number for the line to be updated |
| x       | x-coordinate of the line to be updated    |
| y       | y-coordinate of the line to be updated    |

#### Return Value

None.

#### Exceptions

None.

#### Example

The Resource model used in this appendix does not use LG'upd\_pline.

#### See Also

```
LG'create  
LG'upd_line
```

#### Corresponding real function

```
LG'upd_pline : { lg : (real LINEGRAPH) ref, line :  
int, pattern : int, x : real, y : real} -> unit
```

# Chapter A5

## Version 1.9

### Matrix Chart Functions

#### Version 1.9 Matrix Chart Example

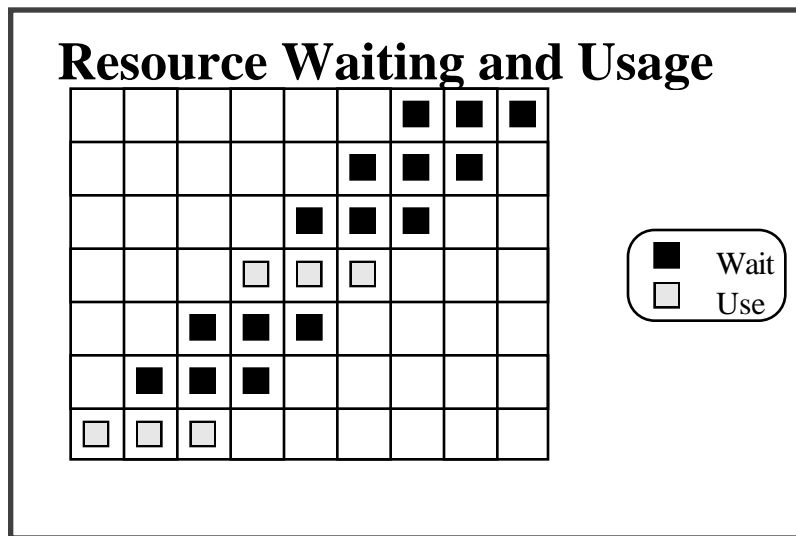


Fig. A5-1: Resource Use Model - Matrix Chart (mc\_resmod)

#### Table of Version 1.9 Matrix Chart Functions

|                  |                     |
|------------------|---------------------|
| <b>Creating</b>  | MC'dec<br>MC'create |
| <b>Updating:</b> |                     |
| Assigning labels | MC'upd_ltag         |
| Appending values | MC'fill<br>MC'write |
| <b>Deleting</b>  | MC'delete           |

### MC'create

Creates a matrix chart.

#### Synopsis

```
MC'create:
  { height : int,      (* Height of the chart *)
    width : int,      (* Width of the chart *)
    x : int,          (* Horizontal location on the page of the
                      upper right-hand corner *)
    y : int,          (* Vertical location on the page of the upper
                      right-hand corner.  *)
    i : int,          (* Maximum number of x-coordinate cells *)
    j : int,          (* Maximum number of y-coordinate cells *)
    pattern : int,    (* Number of cell patterns *)
    legend : bool,    (* Specifies whether or not the chart has a
                      legend identifying line patterns *)
    title : string,   (* Text of the chart title *)
  }
-> MCHART
```

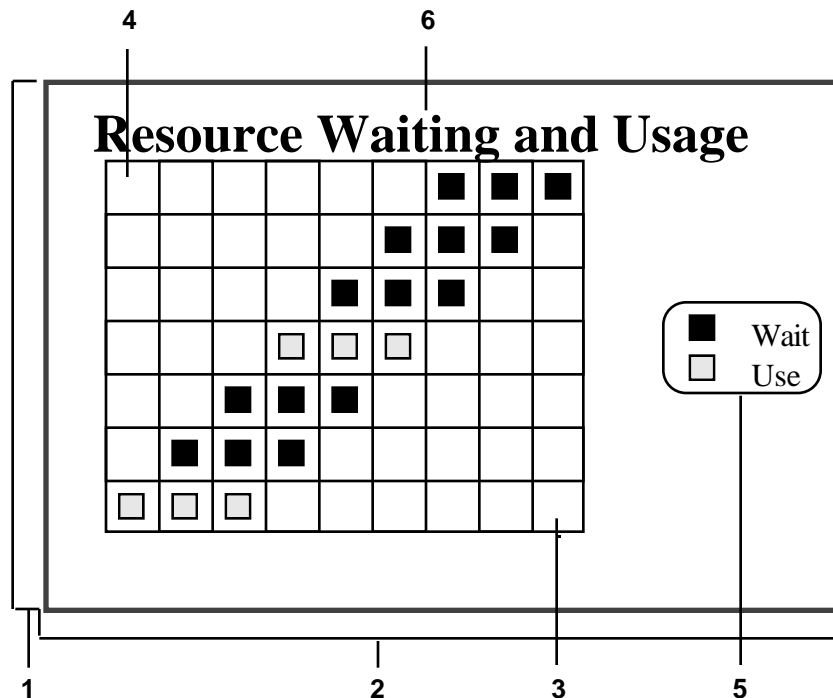
#### Description

Creates an integer matrix chart.

#### Arguments

The Figure Location column in the table below refers to this diagram:





| Argument | Type   | Example    | Figure Location | Comments  |
|----------|--------|------------|-----------------|---|
| height   | int    | 200        | 1               | Height of the chart. The value is in pixels.  |
| width    | int    | 500        | 2               | Width of the chart. The value is in pixels.   |
| x        | int    | 0          | N/A             | Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> . |
| y        | int    | 700        | N/A             | Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .   |
| i        | int    |            | 3               | Maximum number of x-coordinate cells  |
| j        | int    |            | 4               | Maximum number of y-coordinate cells  |
| pattern  | int    |            | 5               | Number of cell patterns   |
| legend   | bool   |            | 5               | Specifies whether or not the chart has a legend identifying line patterns. <code>MC'upd_ltag</code> determines the labels for the patterns.               |
| title    | string | See figure | 6               | Text of the chart title.  |

## Version 1.9 Chart Functions

---

### Return Value

Reference to a matrix chart

### Exceptions

None.

### Example

To create the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
mc_resmod:= MC'create
{height = 200, width = 300, i = ProcCount,
 j = ResCount, legend = true, patterns = 2, x = 0,
 y = 350, title = "Resource Waiting and Usage" };
```

### See Also

```
MC'dec
MC'delete
MC'upd_ltag
```

### Corresponding real function

N/A

### MC'dec

Declares an integer matrix chart.

### Synopsis

```
MC'dec: () -> (int MCHART) ref
```

### Description

Declares a matrix chart. This function has the effect of creating an ML value that can be used together with `MC'create` to create an integer matrix chart.

### Arguments

None.

### Return Value

Reference to a matrix chart.

### Exceptions

None.

### Example

To declare the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
val mc_resmod = MC'dec ();
```

### See Also

`MC'create`

### Corresponding real function

N/A

### MC'delete

Deletes an integer matrix chart.

#### Synopsis

```
MC'delete : ((int MCHART) ref) -> unit
```

#### Description

Deletes an integer matrix chart.

#### Arguments

Reference to a matrix chart

#### Return Value

None.

#### Exceptions

None.

#### Example

To delete the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
MC'delete mc_resmod;
```

#### See Also

`MC'create`

#### Corresponding real function

N/A

### MC'fill

Updates the fill pattern in an integer matrix chart.

### Synopsis

```
MC'fill : { mc : (int MCHART) ref, pattern : int, i :  
int, j : int } -> unit
```

### Description

Updates the fill pattern in an integer matrix chart.

### Arguments

|         |                                    |
|---------|------------------------------------|
| mc      | Reference to a matrix chart        |
| pattern | Fill pattern number                |
| i       | X coordinate of cell to be updated |
| j       | Y coordinate of cell to be updated |

### Return Value

None.

### Exceptions

None.

### Example

To update the fill pattern for the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
MC'fill {mc = mc_resmod, pattern = 1,  
  i = (index'Res res1), j = (index'Proc pid)};  
MC'fill {mc = mc_resmod, pattern = 1,  
  i = (index'Res res2), j = (index'Proc pid)};  
MC'fill {mc = mc_resmod, pattern = 1,  
  i = (index'Res res3), j = (index'Proc pid)};
```

### See Also

MC'create  
MC'write

## Version 1.9 Chart Functions

---

### Corresponding real function

N/A

### MC'upd\_ltag

Updates pattern legend labels on an integer matrix chart.

#### Synopsis

```
MC'upd_ltag : { mc : (int MCHART) ref, tags : string  
list } -> unit
```

#### Description

Updates pattern legend labels on an integer matrix chart.

#### Arguments

|      |   |
|------|---|
| mc   | Reference to a matrix chart                         |
| tags | List of strings giving labels for the fill patterns |

#### Return Value

None.

#### Exceptions

None.

#### Example

The Resource Use model used in this appendix does not use MC'upd\_ltag.

#### See Also

MC'create

#### Corresponding real function

N/A

### MC'write

Updates cell values in an integer matrix chart.

#### Synopsis

```
MC'write : { mc : (int MCHART) ref, i : int, j : int,  
text : string } -> unit
```

#### Description

Updates cell values in an integer matrix chart.

#### Arguments

|      |                                    |
|------|------------------------------------|
| mc   | Reference to a matrix chart        |
| i    | X coordinate of cell to be updated |
| j    | Y coordinate of cell to be updated |
| text | Text for updating cell             |

#### Return Value

None.

#### Exceptions

None.

#### Example

The Resource Use model used in this appendix does not use MC'write.

#### See Also

```
MC'create  
MC'fill
```

#### Corresponding real function

N/A



