Visual HyperScript API Specification

Ucamco Software Product Group
March 2018

Ucamco
Former Barco ETS

March 2018

Correspondence regarding this publication can be sent to:

Ucamco NV
Bijenstraat 19,
B-9051 Gent,
Belgium

For more information:

Our web site: http://www.ucamco.com

E-mail: info@ucamco.com

About Ucamco

Ucamco (formerly Barco ETS) is a market leader in PCB CAM software, photoplotting and direct imaging systems, with a global network of sales and support centers. Headquartered in Ghent, Belgium, Ucamco has over 25 years of ongoing experience in developing and supporting leading-edge photoplotters and front-end tooling solutions for the global PCB industry. Key to this success is the company's uncompromising pursuit of engineering excellence in all its products. Ucamco also owns the IP rights on the Gerber File Format through its acquisition of Gerber Systems Corp. (1998).

Helpdesk

Europe, Middle East, Africa, Latin Amerika	Asia Pacific	North America
Customer Support for Plotters, Software en ManiaBarco AOI Monday - Friday: 08.00 AM - 6.00 PM MET ✆ + 32 9 216 99 00 General support: support@ucamco.com License: license@ucamco.com Java™ HyperTool: hypertool@ucamco.com HyperScript: hyperscript@ucamco.com General information: info@ucamco.com Sales: sales@ucamco.com	Please contact our business partner for your country during support working hours see contacts page	Customer Support for Plotters, Software en ManiaBarco AOI Monday - Friday: 08.00 AM - 6.00 PM Pacific Time Saturday : 10 am to 4 pm Pacific Time +1 949 632 6895 General support: support@ucamco.us License: license@ucamco.us Java™ HyperTool: hypertool@ucamco.com HyperScript: hyperscript@ucamco.com General information: info@ucamco.us

Visual HyperScript API Specification

Classes

- class **Arc**
- class **Line**
- class **Point**
- class **Rectangle**

Functions

- void **abort** (String sInfo)
- void **activate** (String layClass, String laySubclass, int layNum, boolean layAct)
- void **activate** (ObjectList layerID, boolean layAct)
- void **activate** (String layClass, String laySubclass, boolean layAct)
- void **activateAllLayers** ()
- void **activateBottomLayers** (boolean layAct)
- void **activateToggle** ()
- void **activateTopLayers** (boolean layAct)
- void **activityClean** (String sMode)
- void **activityClean** ()
- void **activityRestore** (String sMode)
- void **activityRestore** ()
- void **activityStore** (String sMode)
- void **activityStore** ()
- void **addBreak** (**Point** line_fp, **Point** line_tp)
- void **addBreak** (double line_fx, double line_fy, double line_tx, double line_ty)
- void **addBreak** (**Line** line)
- int **addCFMEEAlignmentPoint** (double point_x, double point_y)
- int **addCFMEEAlignmentPoint** (**Point** point)
- void **addDPF** (String dpf, String layName, String subClass, int layPos, String readable)
- void **addDPFDrill** (String dpf, int layPos, int drillTop, int drillBot)
- void **addDPFExtra** (String dpf, String attach, boolean bNoChecks)
- void **addDPFExtra** (String dpf, String attach)
- void **addDPFLayer** (String dpf, int layPos)
- int **addETMComponentHiPot** (int etmId, int primId, int NetPrim, int NetSecond, String TestVolt, String Duration, String LeakCurrent, String VoltType, String StartVolt, String VoltRise)
- int **addETMComponentHiPot** (int etmId, int primId, double XStart, double YStart, String AccStart, double XEnd, double YEnd, String AccEnd, int NetPrim, int NetSecond, String TestVolt, String Duration, String LeakCurrent, String VoltType, String StartVolt, String VoltRise)
- void **addFault** (String sType, double oRectangle_xmin, double oRectangle_ymin, double oRectangle_xmax, double oRectangle_ymax, String sInfo)
- void **addFault** (String sType, **Rectangle** oRectangle, String sInfo)
- void **addFault** (String sType, **Line** oLine, String sInfo)
- void **addFault** (String sType, double oPt_x, double oPt_y, String sInfo)
- void **addFault** (String sType, **Point** oPt, String sInfo)
- void **addHyperScriptMenuItem** (String sScriptPath, String sMenuItemLabel)
- void **addMaskLayer** (double dThicken)
- void **addObjectAttribute** (String sAttrName)
- void **addObjectAttribute** (String sAttrName, String sAttrValue)
- void **addOptimizedMaskLayer** (double dMinRing, double dMaxRing, double dMaskToCopper, double dMaskToMask, double dBigRing)
- void **addRefPoint** (int iIndex, double pPnt_x, double pPnt_y, boolean bOnAllActiveLay)
- void **addRefPoint** (int iIndex, **Point** pPnt, boolean bOnAllActiveLay)
- void **addShavedMaskLayer** (double dThicken, double dPadToTrack, double dPadToPad)
- void **addTeardrops** (int iMode, double dRelDiam, double dRelDist, double dAbsDiam, double dAbsDist, double dMinClr, int iOnRect, int iOnHoles)
- void **addTeardrops** (int iMode, double dRelDiam, double dRelDist, double dAbsDiam, double dAbsDist, double dMinClr, int iOnRect)
- int **addYsphotechAlignmentPoint** (int region, double point_x, double point_y)

int **addYsphotechAlignmentPoint** (int region, **Point** point)
- void **align_blocks** (double dTolerance, boolean bOnAllLayers)
- void **AmliAddUser** (String sUser, String sPassword, String sAuthorityLevel)
- void **AmliChangeUserPwd** (String sUser, String sPassword, String sNewPassword)
- String **AmliCheckUser** (String sUser, String sPassword)
- void **AmliRemoveUser** (String sUser)
- String **analyzeExternal** (String sExtName)
- void **angle** (double angle)
- double **angle** ()
- void **apeAnamorphicScale** (double dDistanceX, double dDistanceY, boolean bProportional)
- void **apeAnamorphicScale** (double dScaleX, double dScaleY)
- void **apeAttribute** (String name, String value)
- String **apeAttribute** (String name)
- String **apeAttribute** ()
- void **apeCorners** (String sCorners)
- String **apeCorners** ()
- void **apeCreateBox** (int apeNum, double xsize, double ysize, String corners, double xcutoff, double ycutoff)
- void **apeCreateCircle** (int apeNum, double dia)
- void **apeCreateContour** (int apeNum, double stroke)
- void **apeCreateDonut** (int apeNum, double outer, double inner, String kind)
- void **apeCreateOblong** (int apeNum, double xsize, double ysize)
- void **apeCreateOctagon** (int apeNum, double size)
- void **apeCreateRectangle** (int apeNum, double xsize, double ysize)
- void **apeCreateText** (int iApeNum, double dHeight, String sText, double dRotation)
- void **apeCreateText** (int iApeNum, double dHeight, String sText)
- void **apeCreateThermal** (int apeNum, double outer, double inner, double gap, int numGap, double angle, String kind)
- **Rectangle apeEnclosingBox** ()
- int **apeExtlinkCheck** ()
- String **apeExtlinkPath** ()
- String **apeExtlinkPathString** ()
- boolean **apeExtlinkRelative** ()
- int **apeExtlinkStatus** ()
- void **apeGap** (double dGap)
- double **apeGap** ()
- boolean **apeHasPattern** (boolean bUsed)
- void **apeHeight** (double dHeight)
- double **apeHeight** ()
- int **apeIndex** ()
- String **apeInfo** ()
- void **apeInner** (double dInner)
- double **apeInner** ()
- void **apeKind** (String sKind)
- String **apeKind** ()
- int **apeMaxNetNumber** ()
- void **apeMirror** (String sMirror)
- String **apeMirror** ()
- void **apeName** (String sName)
- String **apeName** ()
- void **apeNumber** (int iNumber)
- int **apeNumber** ()
- void **apeNumberGap** (int iNumGap)
- int **apeNumberGap** ()
- int **apeNumberObject** (String sObjectClass)
- int **apeNumberObject** ()
- int **apeNumberRegions** ()
- int **apeNumContours** ()
- void **apeOuter** (double dOuter)
- double **apeOuter** ()
- void **apePattern** (String sPattern)
- String **apePattern** ()

void **apePatternAngle** (double dPatternAngle)
- double **apePatternAngle** ()
- void **apePatternStep** (double dPatternStep)
- double **apePatternStep** ()
- void **apePatternWidth** (double dPatternWidth)
- double **apePatternWidth** ()
- void **apePatternX** (double dX)
- double **apePatternX** ()
- void **apePatternY** (double dY)
- double **apePatternY** ()
- **Rectangle apeRectangle** ()
- void **apeReverse** (boolean bReverse)
- boolean **apeReverse** ()
- void **apeRotation** (double dRotation)
- double **apeRotation** ()
- void **apeScale** (double dScale)
- double **apeScale** ()
- boolean **apeSelection** ()
- **Rectangle apeSelectionEnclosingBox** ()
- String **apeShape** ()
- void **apeSize** (double dSize)
- double **apeSize** ()
- void **apeStartAngle** (double dStartAngle)
- double **apeStartAngle** ()
- void **apeString** (String sString)
- String **apeString** ()
- void **apeStroke** (double dStroke)
- double **apeStroke** ()
- double **apeSurface** ()
- void **apeThickenThin** (double value, boolean keepArcs)
- void **apeWidth** (double dWidth)
- double **apeWidth** ()
- void **apeXCutOff** (double dXCutOff)
- double **apeXCutOff** ()
- void **apeXSize** (double dXSize)
- double **apeXSize** ()
- void **apeYCutOff** (double dYCutOff)
- double **apeYCutOff** ()
- void **apeYSize** (double dYSize)
- double **apeYSize** ()
- int **applyHorns** (String hornType, double minimumClearance, ObjectList params)
- **Arc Arc** (double ptFromX, double ptFromY, double ptToX, double ptToY, double ptCenterX, double ptCenterY, String sSense, String sUnits)
- **Arc Arc** (double ptFromX, double ptFromY, double ptToX, double ptToY, double ptCenterX, double ptCenterY, String sSense)
- **Arc Arc** (**Arc** oArc)
- **Arc Arc** (**Point** ptFrom, **Point** ptTo, **Point** ptCenter, String sSense)
- void **autofixtureBuildFixture** (boolean bFixtureBuild, String sFixture)
- void **autofixtureDo** ()
- void **autofixtureMicroAdjustment** (boolean bMicroAdjustment, int iNbrOfTestPoints, double dTestPointDiameter, double dTestPointShiftEdge, double dTestPointShiftValue, double dTestPointPitch, double dClearanceFactor, double dCenterDiameter)
- void **autofixtureNetlist** (boolean bNetlist, boolean bNetlistBuild, boolean bNetlistExpand)
- void **autofixtureOutput** (boolean bOutput)
- void **autofixtureTestpoints** (boolean bTestPoints, int iLoop, boolean bUseMasks, boolean bProbeSwaping, boolean bHandlePaintedPads, boolean bCircuitryCheck, boolean bFilterCopperAreas, boolean bViaOfSMDs, boolean bDrillsWithoutPad)
- void **autofixtureTestpointsBot** (boolean bPointsBot1, boolean bPointsBot2, boolean bPointsBot3, boolean bPointsBot4, boolean bPointsBot5, boolean bPointsBot6, boolean bPointsBot7)
- void **autofixtureTestpointsTop** (boolean bPointsTop1, boolean bPointsTop2, boolean bPointsTop3, boolean bPointsTop4, boolean bPointsTop5, boolean bPointsTop6, boolean bPointsTop7)

void **blockEdit** ()
- void **blockMultiEdit** ()
- int **BlockReconstruct** ()
- void **boardSnapshot** (boolean graph, String templPath, boolean pio, String pioPath)
- void **buildSubJobs** ()
- void **calculateImpedance** (String sImpConfig, ObjectList parameters)
- boolean **canRead** (ObjectList fileInfo)
- boolean **canWrite** (ObjectList fileInfo)
- void **center** (double center_x, double center_y)
- void **center** (**Point** center)
- **Point center** ()
- void **centerX** (double centerX)
- void **centerY** (double centerY)
- void **chain** ()
- void **chamferJoin** (double pt_x, double pt_y, double disX, double disY)
- void **chamferJoin** (**Point** pt, double disX, double disY)
- void **changeDirection** (double p_x, double p_y)
- void **changeDirection** (**Point** p)
- int **changePrioPlotQueue** (String sRipHost, int iJobId, int iPriority)
- boolean **checkDrillInfo** (boolean bSelNonPlated, boolean bAssignAttributes, boolean bBlocksOnly)
- String **chooseDirPath** (String sTitle, String sStartDir)
- String **chooseDirPath** (String sTitle)
- String **chooseDirPath** ()
- String **chooseFilePath** (String sTitle, String sStartDir, String sFileMask)
- String **chooseFilePath** (String sTitle)
- String **chooseFilePath** (String sStartDir, String sFileMask)
- String **chooseFilePath** ()
- void **cleanApertures** ()
- void **cleanApeTables** ()
- void **cleanETMComponentLayers** (int type)
- void **cleanSubJobs** ()
- void **cleanUfd** (String sUfdName)
- void **cleanUnderBlo** ()
- void **cleanup** (double dReconstructArcs, double dValidateArcs, double dRemoveObsoleteObjects, double dRemoveSmallObjects, double dReconnectObjects, boolean bReconstructArcs, boolean bValidateArcs, boolean bRemoveObsoleteObjects, boolean bRemoveSmallObjects, boolean bReconnectObjects)
- void **clearance** (double clearance)
- double **clearance** ()
- void **clearanceCheckMAT** (double dPadSpread, double dSmdSpread, double dTrackSpread, double dAreaSpread, double dPadPadClearance, double dPadSmdClearance, double dPadTrackClearance, double dPadAreaClearance, double dSmdSmdClearance, double dSmdTrackClearance, double dSmdAreaClearance, double dTrackTrackClearance, double dTrackAreaClearance, double dAreaAreaClearance, boolean bCheckSameNetSpacing, boolean bFastMode, int iShiftMode, double dMinCopper)
- void **clearanceCheckMAT** (double dPadSpread, double dSmdSpread, double dTrackSpread, double dAreaSpread, double dPadPadClearance, double dPadSmdClearance, double dPadTrackClearance, double dPadAreaClearance, double dSmdSmdClearance, double dSmdTrackClearance, double dSmdAreaClearance, double dTrackTrackClearance, double dTrackAreaClearance, double dAreaAreaClearance, boolean bCheckSameNetSpacing, boolean bFastMode, int iShiftMode)
- void **clearanceCheckMAT** (double dPadSpread, double dSmdSpread, double dTrackSpread, double dAreaSpread, double dPadPadClearance, double dPadSmdClearance, double dPadTrackClearance, double dPadAreaClearance, double dSmdSmdClearance, double dSmdTrackClearance, double dSmdAreaClearance, double dTrackTrackClearance, double dTrackAreaClearance, double dAreaAreaClearance, boolean bCheckSameNetSpacing, boolean bFastMode)
- void **clearMessages** ()
- boolean **clipping** (int iClipReference, String sClipSide, double dClipClr, double dMinLineLength, boolean bRounded)
- boolean **clipSilk** (double dClr, double dMinLen)
- void **closeAMLIJobManager** ()
- void **closeAnamorphicScale** ()
- void **closeApeCreator** ()

void **closeApeEditor** ()
- void **closeApertureAttributes** ()
- void **closeApertureManager** ()
- void **closeAttributeEditor** ()
- void **closeAttributeManager** ()
- void **closeAutoDrill** ()
- void **closeAutoDrillEditor** ()
- void **closeAutoFixture** ()
- void **closeBarcode** ()
- void **closeBarcode128** ()
- void **closeBoardAnalyzer** ()
- void **closeBoardSnapshot** ()
- void **closeCalculatorSetup** ()
- void **closeCamtek** ()
- void **closeCheckList** ()
- void **closeCheckListDefineChecklist** ()
- void **closeCheckListDefineSteps** ()
- void **closeClipping** ()
- void **closeColor** ()
- void **closeConnect** ()
- void **closeContourHandling** ()
- void **closeConvertAttributes** ()
- void **closeCopperBalance** ()
- void **closeCopperRepair** ()
- void **closeCoverlayOptimizer** ()
- void **closeCU9000Dialog** ()
- void **closeDatums** ()
- void **closeDistort** ()
- void **closeDrawSlots** ()
- void **closeDRC** ()
- void **closeDrillInfo** ()
- void **closeDrillMap** ()
- void **closeDrillOptimizer** ()
- void **closeDrillRoutSetups** ()
- void **closeDrillTolerance** ()
- void **closeDrillToolManager** ()
- void **closeDsAoi** ()
- void **closeDSAOIialog** ()
- void **closeDsAoiPreview** ()
- void **closeEditingToolbox** ()
- void **closeEditVectorText** ()
- void **closeErrors** ()
- void **closeEtchCompensation** ()
- void **closeExpand** ()
- void **closeExternalLinkManager** ()
- void **closeFiducials** ()
- void **closeFillAngledPattern** ()
- void **closeFillPattern** ()
- void **closeFillVector** ()
- void **closeFlashMaker** ()
- void **closeFlexManager** ()
- void **closeFlipJob** ()
- void **closeFrame** (String sFrameName)
- void **closeGridParameters** ()
- void **closeHiPot** ()
- void **closeImageCompare** ()
- void **closeImpedanceControl** ()
- void **closeImportODBxx** ()
- void **closeInsertContourText** ()
- void **closeInsertVectorText** ()
- void **closeJobDefinition** ()

void **closeJobEdit** ()
- void **closeJobEditor** ()
- void **closeJobEditorOptions** ()
- void **closeJobMerge** ()
- void **closeJobPlaneSetup** ()
- void **closeJobPrint** ()
- void **closeLayerEdit** ()
- void **closeLegendOptimizer** ()
- void **closeLoadCheckList** ()
- void **closeMagnifier** ()
- void **closeMarkupAssistant** ()
- void **closeMessages** ()
- void **closeMLIOutput** ()
- void **closeModels** ()
- void **closeNetCompare** ()
- void **closeNonFunctionalPad** ()
- void **closeNumbers** ()
- void **closeObjectAttributes** ()
- void **closeObjectCompare** ()
- void **closeOutputAccumatch** ()
- void **closeOutputAOI** ()
- void **closeOutputCAD** ()
- void **closeOutputCamtek** ()
- void **closeOutputDrillRout** ()
- void **closeOutputDsDi** ()
- void **closeOutputDsDiPreview** ()
- void **closeOutputNetlist** ()
- void **closeOutputOrbot** ()
- void **closeOutputSapphire** ()
- void **closeOutputScoring** ()
- void **closeOutputSmartArgos** ()
- void **closeOutputTrackscan** ()
- void **closeOutputUxpAutomanager** ()
- void **closeOutputUxpEtec** ()
- void **closePanelFramesCoupons** ()
- void **closePanelLinks** ()
- void **closePanelPlus** ()
- void **closePanelReproduce** ()
- void **closePanelSetup** ()
- void **closePanelStepRepeat** ()
- void **closePlaneAdjuster** ()
- void **closePlotParameters** ()
- void **closePPMonitor** ()
- void **closeQueryNet** ()
- void **closeQueryObject** ()
- void **closeReferencePoints** ()
- void **closeRegister** ()
- void **closeRemoveAttributes** ()
- void **closeRepair** ()
- void **closeRoutManager** ()
- void **closeRoutManagerCleanUp** ()
- void **closeRoutManagerDimensioning** ()
- void **closeRoutManagerEditor** ()
- void **closeRoutManagerTools** ()
- void **closeSaveLayout** ()
- void **closeSecureEtchCompensation** ()
- void **closeSelections** ()
- void **closeSetupOptions** ()
- void **closeSetupSave** ()
- void **closeShavePads** ()
- void **closeSignalLayerAdjuster** ()

void **closeSignalLayerAdjusterAssistant** ()
- void **closeSilkOptimizer** ()
- void **closeSmartCamtek** ()
- void **closeSmartDRC** ()
- void **closeSmartFix** ()
- void **closeSmartplot** ()
- void **closeSmartSR** ()
- void **closeSmartStart** ()
- void **closeSoldermask** ()
- void **closeSoldermaskOptimizer** ()
- void **closeTearDrop** ()
- void **closeTechnicalAnalyzer** ()
- void **closeTestpointEdit** ()
- void **closeToolbarManager** ()
- void **closeToolbars** ()
- void **closeTransformObjects** ()
- void **closeTransformObjectsBGAPads** ()
- void **closeTransformObjectsBGATracks** ()
- void **closeTransformObjectsEdit** ()
- void **closeTransformObjectsRescale** ()
- void **closeUcamDbEditor** ()
- void **closeUndoRedoDetails** ()
- void **closeUTest** ()
- void **closeUtestUtilities** ()
- void **closeValidateLayer** ()
- void **closeVectorTextFont** ()
- void **closeVerifyArcsDraws** ()
- void **closeViewGuide** ()
- void **colorAll** (String exclSubClass, boolean bKeepLayActivity)
- void **colorAll** (String exclSubClass)
- void **compareImage** (String reference, boolean bAutoAlign, double missingTol, double exceedingTol, int iErrorAccuracy, ObjectList revPolArr, int compSelMode)
- void **compareImage** (String reference, boolean bAutoAlign, double missingTol, double exceedingTol, int iErrorAccuracy, ObjectList revPolArr)
- void **compareImage** (double missingTol, double exceedingTol, int iErrorAccuracy, boolean bRevPolarityCur, boolean bRevPolarityRef, int compSelMode)
- void **compareImage** (double missingTol, double exceedingTol, int iErrorAccuracy, boolean bRevPolarityCur, boolean bRevPolarityRef)
- void **compareNet** (int iMode, boolean bCheckFlash)
- void **compareNet** (int iMode, boolean bCheckFlash, String sReferenceFile, boolean bPanelize)
- void **compareNet** (int iMode, boolean bCheckFlash, boolean bIgnoreOutline, double dOutlineMargin, boolean bIgnoreNPTHPads, double dNPTHExpandMargin, String sReferenceFile, boolean bPanelize)
- void **compareNet** (boolean bShorts, boolean bOpens, boolean bLostElements, boolean bDoubleNet, boolean bNotCovered, boolean bCheckFlash, boolean bIgnoreOutline, double dOutlineMargin, boolean bIgnoreNPTHPads, double dNPTHExpandMargin, String sReferenceFile, boolean bPanelize, boolean bBuildNetlist)
- void **compareObjects** (String referenceJob, double xTol, double yTol, boolean bWindow, boolean bObjMoved, boolean bObjAdded, boolean bObjNet, boolean bApeShape, boolean bApeSize, boolean bApeOrder)
- void **compensate** (String sSense, double dis)
- void **complexEdit** ()
- int **connectPadTrack** (double dActiveRadius, double dSnapRadius, boolean bUseNetlist)
- void **connectTracks** ()
- boolean **contourizeBitmap** (int iPpi, double dMargin, double dDxdy, double dDx, double dDy)
- void **contourizeExact** ()
- void **contourizeExactAperture** ()
- boolean **contourizePatterns** (int iPpi)
- boolean **contourizePatternsinJob** (int iPpi)
- void **contourThickenThin** (double value, boolean bKeepArcs)
- boolean **convertGar** (String sInputFile, String sGarFile, String sOutputFile)
- void **copperBalancePad** (double dMinClrToCopper, double dMinClrToBoard, double dMinConSurface,

String sFillPattern, double dPatternClr, double dApeSize, String sApeShape)
- void **copperBalanceSolid** (double dMinClrToCopper, double dMinClrToBoard, double dMinConSurface, double dApeSize)
- void **copperBalanceTrack** (double dMinClrToCopper, double dMinClrToBoard, double dMinConSurface, String sLineStyle, double dPatternClr, double dApeSize, double dRotation)
- void **copperCount** (String sOpt)
- void **copperRepair** (String sOpt, double smallerThan, double minSize, double expand)
- void **copy** (double pt_x, double pt_y)
- void **copy** (**Point** pt)
- void **copyOutline** (double refPoint_x, double refPoint_y, double offset_x, double offset_y, double rotation)
- void **copyOutline** (**Point** refPoint, **Point** offset, double rotation)
- void **copyToClipboard** ()
- void **coreInfo** ()
- int **countAmbiguousContours** ()
- int **countAmbiguousContoursOnLayer** ()
- int **countInvalidArcs** ()
- int **countInvalidArcsOnLayer** ()
- int **countInvalidDraws** ()
- int **countInvalidDrawsOnLayer** ()
- int **countOpenContours** ()
- int **countOpenContoursOnLayer** ()
- int **countOverlapContours** ()
- int **countOverlapContoursOnLayer** ()
- int **countUndefinedApertures** ()
- int **countUndefinedAperturesOnLayer** ()
- int **countZeroDrawsArcs** (double dMaxLength, boolean bFunctional, boolean bNonFunctional)
- int **countZeroDrawsArcsOnLayer** (double dMaxLength, boolean bFunctional, boolean bNonFunctional)
- void **countZeroLengthDrawsArcs** (double dMaxLength, boolean bFunctional, boolean bNonFunctional)
- void **createAperture** (int iApeNum, String sApeName, String sApeDef, ObjectList attrArray)
- boolean **createBarcode128** (double dHeight, double dNarrowX, String sValue)
- boolean **createBarcode39** (double dHeight, double dNarrowX, double dRatio, String sValue)
- boolean **createBarcodeInterleaved25** (double dHeight, double dNarrowX, String sValue)
- void **createBlockAperture** (int iApeNum, String sApeName, String sApeDef, ObjectList attrArray, int iMode, boolean bWithCenter, double pt_x, double pt_y, String sExtFile)
- void **createBlockAperture** (int iApeNum, String sApeName, String sApeDef, ObjectList attrArray, int iMode, boolean bWithCenter, **Point** pt, String sExtFile)
- void **createComplexAperture** (int iApeNum, String sApeName, String sApeDef, ObjectList attrArray, boolean bUseRegion, boolean bWithCenter, double pt_x, double pt_y)
- void **createComplexAperture** (int iApeNum, String sApeName, String sApeDef, ObjectList attrArray, boolean bUseRegion, boolean bWithCenter, **Point** pt)
- void **createDataMatrix** (String sTextToEncode, String sMode, String sFormat, double dDotSize, double dRotation, String sMirror, double dClearance, boolean bReverse)
- void **createDrill** (String layName, String subClass, int from, int to)
- void **createDrill** (int laynum, int drillFrom, int drillTo)
- void **createExtra** (String layName, String subClass, String attach, int index)
- void **createExtra** (String layName, String subClass, String attach)
- void **createExtra** (String attach)
- void **createLayer** (String layName, String subClass, int layPos, String readable)
- void **createLayer** (int laynum)
- void **createQRCode** (String sCode, double dDotSize, double dAngle, String sMirror, double dClr, double dLabelClr, String sLabelPos, boolean bMicroQR, boolean bReverse)
- void **createQRCode** (String sCode, double dDotSize, double dAngle, String sMirror, double dClr, String sLabel, double dLabelClr, String sLabelPos, boolean bMicroQR, boolean bReverse)
- void **createQRCode** (String sCode, double dDotSize, double dAngle, String sMirror, double dClr, boolean bMicroQR, boolean bReverse)
- void **createSubJob** (int from, int to, int selectedSubJob, int selectedLevel)
- void **createVoronoiDiagram** (int iEdgeTypes)
- void **createVoronoiDiagram** (int iEdgeTypes, boolean bExpandArcs)
- void **createVoronoiEdgesExtFile** (boolean bExpandArcs, String sFilePath, String sOptions)
- void **CU9000ApplyPlotstamps** (ObjectList plotstamps)
- boolean **CU9000CheckPlotstamps** ()

boolean **CU9000DetectAutoAreas** (String resultLayerName, String referenceLayerName, double margin)
- boolean **CU9000DetectExactAreas** (String resultLayerName, int blockMode, String pcbName, String referenceLayerName, double margin, double outline)
- int **CU9000DetectGlobalAlignment** (String sAPRefLayerName)
- int **CU9000DetectGlobalAlignment** ()
- int **CU9000DetectLocalAlignmentPoints** (String sAPRefLayerName)
- int **CU9000DetectLocalAlignmentPoints** ()
- boolean **CU9000DetectRectangularAreas** (String resultLayerName, int blockMode, String pcbName, String referenceLayerName, double margin)
- ObjectList **CU9000GetPlotstamps** ()
- void **CU9000GUIApply** ()
- void **CU9000GUILoadAlignment** (String sAlignmentPath)
- void **CU9000GUILoadBrd** (String sBrdPath)
- void **CU9000GUILoadRgi** (String sRgiPath)
- void **CU9000GUISaveAlignment** (String sAlignmentPath)
- boolean **CU9000LoadBoardSetup** (String path)
- boolean **CU9000LoadResistSetup** (String path)
- boolean **CU9000LoadResources** (String sPropertiesPath, String sPropertiesName, String sConversionFileName)
- ObjectList **CU9000OrderPlotstamps** (Object[] plotstamps, String sLevel, String sAtLevel, String sStart, String sOrder)
- boolean **CU9000Output** (String machine)
- void **CU9000SaveBPIs** ()
- void **CU9000SaveLocalAlignmentPoints** (String sOutputFilePath)
- void **CU9000SaveLocalAlignmentPoints** (String sOutputFilePath, boolean bShareAlignmentMarks)
- boolean **CU9000SetParameters** (String xmlFile)
- void **cutToClipboard** ()
- void **dbBoolean** (String dbKey, Boolean bValue)
- boolean **dbBoolean** (String dbKey)
- boolean **dbBooleanDef** (String dbKey, boolean bDefault)
- void **dbDouble** (String dbKey, Boolean dValue)
- double **dbDouble** (String dbKey)
- double **dbDoubleDef** (String dbKey, double dDefault)
- void **dbInteger** (String dbKey, Integer iValue)
- int **dbInteger** (String dbKey)
- int **dbIntegerDef** (String dbKey, int iDefault)
- void **dbPath** (String dbKey, String sPath)
- String **dbPath** (String dbKey)
- String **dbPathDef** (String dbKey, String sDefault)
- void **dbString** (String dbKey, String sValue)
- String **dbString** (String dbKey)
- String **dbStringDef** (String dbKey, String sDefault)
- void **dbUnitValue** (String dbKey, String sValue)
- void **dbUnitValue** (String dbKey, Double dValue)
- double **dbUnitValue** (String dbKey)
- double **dbUnitValueDef** (String dbKey, double dDefault)
- double **dbUnitValueDef** (String dbKey, String sDefault)
- void **defaultOrder** ()
- void **defineFirst** (double p_x, double p_y)
- void **defineFirst** (**Point** p)
- void **defineGroup** (double p_x, double p_y, int iGroupNumber)
- void **defineGroup** (**Point** p, int iGroupNumber)
- void **defineSelectedGroup** ()
- void **delete** ()
- void **deleteAllCFMEEAlignmentPoints** ()
- void **deleteAllRefPoints** (boolean bOnAllActiveLay)
- void **deleteAllYsphotechAlignmentPoints** (int region)
- void **deleteAperture** ()
- void **deleteApertureAttribute** (String sAttributeName)
- void **deleteCFMEEAlignmentPoint** (int point)
- void **deleteDouble** ()

void **deleteLayerByClass** (String className, String subClass, int num, String side)
- void **deleteLayersByActivation** (boolean active)
- void **deleteLayersByName** (String layName)
- void **deleteLayersByNames** (String layNames)
- void **deleteLayersByPlane** (int plane)
- void **deleteRefPoint** (int iIndex, boolean bOnAllActiveLay)
- void **deleteRefPoints** (ObjectList Indexes, boolean bOnAllActiveLay)
- void **deleteSubJob** (int index, int level)
- void **deleteTrueObjects** ()
- void **deleteWithApe** ()
- void **deleteWorkspace** (String sWorkspaceName)
- void **deleteYsphotechAlignmentPoint** (int region, int point)
- void **deleteZeroLengthDraws** (double dMaxLength, boolean bFunctional, boolean bNonFunctional)
- void **deselectAll** ()
- void **deselectAllApertures** ()
- void **deselectAperture** (ObjectList apeIndexArray)
- void **deselectAperture** ()
- void **deselectObjectAttribute** (String sAttrName, String sAttrValue)
- void **deselectObjectAttribute** (String sAttrName)
- void **deselectObjectByAttribute** (String sAttrName, String sAttrValue)
- void **deselectObjectByAttribute** (String sAttrName)
- boolean **detectPCBOutlines** (String sLayName, String sParams)
- boolean **DetectPlaceHolders** (String sHandlerName, String sParams)
- boolean **DetectPlaceHoldersAtLayer** (String sHandlerName, String sParams)
- void **dimensioning** (String sType, double dApertureSize, ObjectList oPoints, boolean bShowErrors, double dArrowHeadWidth, double dArrowHeadHeight, double dRuleToElement, double dRuleToDimLine, double dTextToDimLine, double dTextHeight, double dTextWidth, double dTolerancePos, double dToleranceNeg, double dToleranceScale, int iFormat, boolean bProjectionHorizontal, boolean bProjectionVertical, String sFontName, int iFontStyle, int iFontSize, String sLabel)
- String **direction** ()
- void **direction** (String sDirection)
- void **distance** (double distance)
- double **distance** ()
- boolean **distort** (double x, double y, double pCenter_x, double pCenter_y)
- boolean **distort** (double x, double y, **Point** pCenter)
- boolean **distort** (double x, double y)
- void **doActiveFunction** ()
- void **doCancelActiveFunction** ()
- void **doCopy** (double offset_x, double offset_y)
- void **doCopy** (**Point** offset)
- void **doMove** (double offset_x, double offset_y)
- void **doMove** (**Point** offset)
- void **doOption** (String sOption)
- String **doOption** ()
- void **doRemoveAttribute** (boolean jobAttr, boolean layAttr, boolean apeAttr, boolean objAttr)
- void **drag** (double clickp_x, double clickp_y, double dRadius, double offset_x, double offset_y, double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax)
- void **drag** (**Point** clickp, double dRadius, **Point** offset, **Rectangle** rect)
- void **drag** (double clickp_x, double clickp_y, double dRadius, double offset_x, double offset_y)
- void **drag** (**Point** clickp, double dRadius, **Point** offset)
- void **dragAngle** (double pt_x, double pt_y, double dRadius, double dist, double mlen, boolean bUseLimit)
- void **dragAngle** (**Point** pt, double dRadius, double dist, double mlen, boolean bUseLimit)
- void **dragAngle** (double pt_x, double pt_y, double dRadius, double dist, double mlen)
- void **dragAngle** (**Point** pt, double dRadius, double dist, double mlen)
- void **dragLayer** (String prevClass, String newClass, int prevPosition, int newPosition, boolean duplicate)
- **Line dragLine** (String sLabel)
- **Rectangle dragRectangle** (String sLabel)
- void **drawLastPlanesInFront** (boolean bDo)
- void **drawSlots** (String sDPFApeRef, double dTolerance, String sDPFSlotApe)
- void **drawSlotsSelect** (String sDPFApe, double dTolerance)
- void **drillMapReplace** (String sSymbolFilePath, ObjectList oMappingTable)

void **DSAOIAlignmentApply** ()
- void **DSAOIAlignmentDetect** (String sMachineType, String sObjectRestrictions, boolean bPositive, boolean bNegative, double dMinimumSize, double dMaximumSize)
- boolean **DSAOIApply** ()
- void **DSAOIAreaDetection** (boolean detectImportantLine, boolean detectImportantSpace, boolean detectImportantClearance, boolean detectImportantDrill, boolean detectImportantFutureDrill, boolean detectImportantMaskOpenings, boolean detectProhibitCopper, boolean detectProhibitSpace, boolean detectProhibitNonfunctionalCopper, ObjectList importantLineWidth, Object[] importantSpaceWidth, double importantClearanceWidth, double importantDrillSpreadValue, double prohibitLineWidth, double prohibitSpaceWidth, double minSliverSize, String pathStrategy, boolean mergeOutput, boolean clipProhibitWithImportant, boolean outputNormalAreas, boolean maskPolarity, boolean paintedArea, double paintedAreaValue, String PCBName, boolean outputDrillBinary)
- void **DSAOIAreaDetection** (boolean detectImportantLine, boolean detectImportantSpace, boolean detectImportantClearance, boolean detectImportantDrill, boolean detectImportantFutureDrill, boolean detectImportantMaskOpenings, boolean detectProhibitCopper, boolean detectProhibitSpace, boolean detectProhibitNonfunctionalCopper, ObjectList importantLineWidth, Object[] importantSpaceWidth, double importantClearanceWidth, double importantDrillSpreadValue, double prohibitLineWidth, double prohibitSpaceWidth, double minSliverSize, String pathStrategy, boolean mergeOutput, boolean clipProhibitWithImportant, boolean outputNormalAreas, boolean maskPolarity, String PCBName, boolean outputDrillBinary)
- void **DSAOIAreaDetection** (boolean detectImportantLine, boolean detectImportantSpace, boolean detectImportantClearance, boolean detectImportantDrill, boolean detectImportantFutureDrill, boolean detectImportantMaskOpenings, boolean detectProhibitCopper, boolean detectProhibitSpace, boolean detectProhibitNonfunctionalCopper, ObjectList importantLineWidth, Object[] importantSpaceWidth, double importantClearanceWidth, double importantDrillSpreadValue, double prohibitLineWidth, double prohibitSpaceWidth, double minSliverSize, String pathStrategy, boolean mergeOutput, boolean clipProhibitWithImportant, boolean outputNormalAreas, boolean maskPolarity, String PCBName)
- void **DSAOIAreaDetection** (boolean detectImportantLine, boolean detectImportantSpace, boolean detectImportantClearance, boolean detectImportantDrill, boolean detectImportantFutureDrill, boolean detectImportantMaskOpenings, boolean detectProhibitCopper, boolean detectProhibitSpace, boolean detectProhibitNonfunctionalCopper, ObjectList importantLineWidth, double importantSpaceWidth, double importantClearanceWidth, double importantDrillSpreadValue, double prohibitLineWidth, double prohibitSpaceWidth, double minSliverSize, String pathStrategy, boolean mergeOutput, boolean clipProhibitWithImportant, boolean outputNormalAreas, boolean maskPolarity, String PCBName)
- void **DSAOIAreasApply** ()
- void **DSAOIDetectRectangularArea** (double dMargin, String sBlockMode, String sPCBName, String sReferenceLayerName, boolean bIsSingleLevel, boolean bIsInspection, boolean bIsPmiP1, boolean bIsPmiP2, boolean bIsDrcP1, boolean bIsDrcP2)
- void **DSAOILayerList** ()
- void **DSAOILayerListAreaApply** ()
- void **DSAOILayerListAreaOutput** ()
- void **DSAOILayerListAreaSelect** ()
- void **DSAOILayerListGroupValue** (String sNewValue)
- void **DSAOILayerListRowDeselect** (int iIndexFrom, int iIndexTo)
- void **DSAOILayerListRowDeselect** (int iRow)
- void **DSAOILayerListRowSelect** (int iIndexFrom, int iIndexTo)
- void **DSAOILayerListRowSelect** (int iRow)
- boolean **DSAOILoadLayerListProfile** (String pro)
- boolean **DSAOIOutput** ()
- void **DSAOIpinpointDetection** (double dStep, double dInfinity, String sOutputFilePath)
- void **DSAOIpinpointDetection** (double dStep, double dInfinity, String sPCBName, String sOutputFilePath, String sLocation)
- void **DSAOIPositionApply** ()
- void **DSAOISetApplyToBackLayers** (boolean bValue)
- void **DSAOISetApplyToFrontLayers** (boolean bValue)
- void **DTMCalculate** (String sPlatingType, String sToleranceScript)
- boolean **DTMCreateSymbolDrawing** ()
- boolean **DTMLoadData** ()
- void **DTMLoadToleranceFile** (String sToleranceFileName)
- void **DTMRemoveAttributes** ()
- void **DTMSaveDataToAttributes** (String sJobName)

void **DTMUpdateDPF** (String sJobName)
- void **duplicateAperture** ()
- void **duplicateLayer** (String layName, String newName)
- void **dwAnnotate** (String annotationLayerName, String sChipID)
- void **dwAnnotate** (String annotationLayerName)
- void **dwApplyTransform** (String sResultFilePath)
- void **editAperture** (int iApeNum, String sApeName, String sApeDef)
- void **emptyClipboard** ()
- void **enlargePads** (String absRel, double aVal, boolean bExclcon, boolean bUseBGA)
- String **envString** (String name)
- void **equalizeTrackSpace** ()
- void **etchCompensation** (boolean bUseExcludeAreas, boolean bCreateLayerBackup, boolean bShowLayerBackup, int iOutStyle, boolean bAsymmetricPadTrackComp, ObjectList arrPrefOffset)
- void **etchCompensation** (boolean bUseExcludeAreas, boolean bUseCompensateAreas1, boolean bUseCompensateAreas2, boolean bCreateLayerBackup, boolean bShowLayerBackup, int iOutStyle, boolean bAsymmetricPadTrackComp, ObjectList arrPrefOffset, Object[] arrCompLay1, Object[] arrCompLay2)
- void **expandArcs** ()
- void **expandBlock** ()
- void **expandNibble** (double overlapValue, double pitchValue, boolean useOverlap)
- void **expandText** ()
- void **expandTrueObjects** ()
- void **expandVtx** ()
- void **externalLinkManagerCheck** ()
- void **filletJoin** (double pt_x, double pt_y, double dis)
- void **filletJoin** (**Point** pt, double dis)
- void **fillPolygon** (boolean bDirection)
- void **fillPolygonCCW** ()
- void **fillPolygonCW** ()
- void **fillWithAngledPattern** (String shape, double size, double pitch, double angle)
- void **fillWithPatternPads** (String sKind, boolean bKeepEdge, double pGridOrigin_x, double pGridOrigin_y, double pGridStep_x, double pGridStep_y)
- void **fillWithPatternPads** (String sKind, boolean bKeepEdge, **Point** pGridOrigin, **Point** pGridStep)
- void **fillWithPatternStarburst** (int iSegments, String sKind, int dBlack, boolean bWithCenter, double pCenter_x, double pCenter_y, boolean bKeepEdge, double dEdgeWith)
- void **fillWithPatternStarburst** (int iSegments, String sKind, int dBlack, boolean bWithCenter, **Point** pCenter, boolean bKeepEdge, double dEdgeWith)
- void **fillWithPatternTracks** (String sPattern, double dStep, double dWidth, double dRotation, boolean bKeepEdge)
- int **fillWithVectors** (double dOverlap, double dDiameter, int iApeCount, int iApeNum, String sFillOpt)
- void **findSections** (String szOptions)
- void **findSlots** ()
- int **findStandardShape** (double dTolerance, String szOpt, String szAction)
- void **flashMakerDeleteComplex** ()
- void **flashMakerDeselectComplex** ()
- void **flashMakerFind** ()
- void **flashMakerFindStandardShapes** (Uxjob oJob)
- void **flashMakerReplace** ()
- void **flashMakerReplaceStandardShapes** (Uxjob oJob)
- void **flashMakerSetup** (double minCutoff, double minSize, double maxSize, boolean useTol, double tol, boolean useMask, boolean deselNonModel)
- void **flashMakerSetup** (double minCutoff, double minSize, double maxSize, boolean useTol, double tol, boolean useMask, boolean completelyFree, boolean deselNonModel)
- void **flipJob** (String mirror, boolean bFlipBuildup, boolean bFlipAttachNone, String suffix)
- void **forEachApe** (String sType)
- void **forEachApe** ()
- void **forEachArc** ()
- void **forEachDraw** ()
- void **forEachDrill** (String sSubClass)
- void **forEachDrill** ()
- void **forEachExtra** (String sSubClass, String sAttach)

void **forEachExtra** (String sSubClass)
- void **forEachExtra** ()
- void **forEachFlash** ()
- void **forEachI8Job** (String serverName, String dbname, String username, String password, String context, String i8path)
- void **forEachInRectangle** (**Rectangle** rect, boolean opt)
- void **forEachInRectangle** (**Rectangle** rect)
- Object **forEachItem** (ObjectList items)
- void **forEachJobNet** ()
- void **forEachLayer** (String sClass, String sSubClass, String sAttach)
- void **forEachLayer** (String sClass, String sSubClass)
- void **forEachLayer** (String sClass)
- void **forEachLayer** ()
- void **forEachLayerNet** ()
- void **forEachNet** (int iNet)
- void **forEachObject** (String sClass)
- void **forEachObject** ()
- void **forEachPEInputJob** ()
- void **forEachPEPanelJob** ()
- void **forEachPESolution** ()
- void **forEachRegion** ()
- void **forEachSignal** (String sSubClass)
- void **forEachSignal** ()
- void **forEachVtxt** ()
- int **GDSII_outLayer** (String filename)
- int **GDSII_outLayer** (String filename, String options)
- boolean **generateContours** (double dGap, double dOverlap)
- boolean **generateContoursOnLayer** (double dGap, double dOverlap)
- ObjectList **getAttrCategories** ()
- ObjectList **getAttrNames** (String sCategory)
- ObjectList **getAttrValues** (String sAttributeName)
- int **getCount** (String sType)
- ObjectList **getFaultTypes** ()
- String **getFileLastModified** (ObjectList fileInfo)
- String **getFileName** (ObjectList fileInfo)
- String **getFileParent** (ObjectList fileInfo)
- long **getFileSize** (ObjectList fileInfo)
- Ulayer **getLayer** (ObjectList layerID)
- ObjectList **getLayerNames** ()
- ObjectList **getLayers** ()
- **Rectangle getLocationOnScreen** (String sFrameName)
- ObjectList **getMode** ()
- ObjectList **getNetAttrNames** ()
- ObjectList **getNetNames** ()
- int **getNetNumberByClick** (double pt_x, double pt_y)
- int **getNetNumberByClick** (**Point** pt)
- Ulayer **getNextLayer** ()
- ObjectList **getODBxxSteps** (String sPath)
- int **getPlotParam** (String sKey, int iDefValue)
- double **getPlotParam** (String sKey, double dDefValue)
- String **getPlotParam** (String sKey, String sDefValue)
- void **grabWidget** ()
- void **gridAlign** (double dStep)
- void **gridCross** (boolean bCross)
- boolean **gridCross** ()
- void **gridOrigin** (double ptOrigin_x, double ptOrigin_y)
- void **gridOrigin** (**Point** ptOrigin)
- **Point gridOrigin** ()
- void **gridOutline** (double refPoint_x, double refPoint_y, double offset_x, double offset_y, int repeatX, int repeatY)
- void **gridOutline** (**Point** refPoint, **Point** offset, int repeatX, int repeatY)

void **gridStep** (double dStepX, double dStepY)
- **Point gridStep** ()
- double **gridStepX** ()
- double **gridStepY** ()
- void **gridVisible** (boolean bVisible)
- boolean **gridVisible** ()
- void **groupApeBy** (String spec)
- void **groupApertureDefinitions** ()
- void **groupApertureNumbers** ()
- void **groupAperturesByPolarity** ()
- double **groupUFD** (double dDistance)
- void **groupUFD** ()
- void **helpOnContext** ()
- void **hideAll** ()
- void **hideBlockStructure** ()
- void **HitachiSpotDiameterCompensation** (double dOffset, double dArcExpandMarginOverrideMicrons, int iMode, int iFastMode, boolean bAmSkipFlag, boolean bChangePolarity)
- void **HitachiSpotDiameterCompensation** (double dOffset, double dArcExpandMarginOverrideMicrons, int iMode, boolean bFastMode, boolean bAmSkipFlag, boolean bChangePolarity)
- String **i8Jobarticleid** ()
- String **i8JobBoardid** ()
- String **i8JobCustomer** ()
- boolean **i8Jobdelete** ()
- int **i8JobDuration** ()
- Date **i8JobFinishtime** ()
- int **i8JobFullduration** ()
- int **i8JobId** ()
- String **i8JobLocation** ()
- int **i8JobPriority** ()
- String **i8JobProgress** ()
- int **i8JobQueueposition** ()
- Date **i8JobStarttime** ()
- Date **i8JobSubmittime** ()
- void **importEpc** (String sPath)
- int **importExternal** (String sExtName, String sWheName, String sLanguage, boolean bKeepExtension, String sLayClass, String sLayAtt, String sStatus, String sWheLang, boolean bAnalyzed, String sWheFile, int iLocale)
- int **importExternal** (String sExtName, String sWheName, String sLanguage, boolean bKeepExtension)
- int **importExternal** (String sExtName)
- void **importFile** (String sScriptPath)
- void **importGwk** (String sPath)
- String **importHeptaCSV** (String sCSVFile, String sDXFFile, String sOptions)
- void **importHouei** (String sPath)
- void **importIpc** (String sPath, String sVersion)
- void **importIPC2581** (String sPath, String sStep, String sLayer)
- void **importODBxx** (String sPath, String sStep, String sLayer, ObjectList oReplaceCodeMap)
- void **importODBxx** (String sPath, String sStep, String sLayer)
- void **importODBxx** (String sPath, String sStep, ObjectList oReplaceCodeMap)
- void **importODBxx** (String sPath, String sStep)
- void **importPolarBuildup** (String sPolarFilePath)
- void **importScript** (String sScript)
- void **importWf** (String sPath)
- void **innerCopperCount** (boolean bUseMask, boolean bConfirmMaskUsage)
- void **innerCopperCount** ()
- void **insertAperture** (boolean bBefore, String sSrcLayer, ObjectList srcApeIndex)
- int **insertArc** (double arc_fx, double arc_fy, double arc_tx, double arc_ty, double arc_cx, double arc_cy, String arc_sense, int iNet, String sSelection)
- int **insertArc** (**Arc** arc, int iNet, String sSelection)
- void **insertArc3Point** (double pnt1_x, double pnt1_y, double pnt2_x, double pnt2_y, double pnt3_x, double pnt3_y)
- void **insertArc3Point** (**Point** pnt1, **Point** pnt2, **Point** pnt3)

void **insertArc3Point** (double pnt1_x, double pnt1_y, double pnt2_x, double pnt2_y, double pnt3_x, double pnt3_y, int iNet, String sSelection)

- void **insertArc3Point** (**Point** pnt1, **Point** pnt2, **Point** pnt3, int iNet, String sSelection)
- void **insertArcCenterStart** (double pntCenter_x, double pntCenter_y, double pntFrom_x, double pntFrom_y, double pntTo_x, double pntTo_y, String sDirection)
- void **insertArcCenterStart** (**Point** pntCenter, **Point** pntFrom, **Point** pntTo, String sDirection)
- void **insertArcCenterStart** (double pntCenter_x, double pntCenter_y, double pntFrom_x, double pntFrom_y, double pntTo_x, double pntTo_y, String sDirection, int iNet, String sSelection)
- void **insertArcCenterStart** (**Point** pntCenter, **Point** pntFrom, **Point** pntTo, String sDirection, int iNet, String sSelection)
- boolean **insertArcConcentric** (boolean bSelection, ObjectList oArcs)
- void **insertBreak** (**Point** line_fp, **Point** line_tp)
- void **insertBreak** (double line_fx, double line_fy, double line_tx, double line_ty)
- void **insertBreak** (**Line** line)
- void **insertContourText** (double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax, String sText, String sFontName, int iFontStyle, String sMirror, boolean bReverse, boolean bAllowDistortion, String sSelection)
- void **insertContourText** (**Rectangle** rect, String sText, String sFontName, int iFontStyle, String sMirror, boolean bReverse, boolean bAllowDistortion, String sSelection)
- void **insertCopper** (int number, String attach, String material, double thickness, String reference, double tolerance, String supplier)
- void **insertCore** (int layNum, boolean matTop, boolean matBot, String material, String topMaterial, String botMaterial, double thickness, double topThickness, double botThickness, String reference, double tolerance, double erConstant, String supplier, ObjectList attrNames, Object[] attrValues, boolean revInsert)
- void **insertCore** (int layNum, boolean matTop, boolean matBot, String material, String topMaterial, String botMaterial, double thickness, double topThickness, double botThickness, String reference, double tolerance, double erConstant, String supplier, ObjectList attrNames, Object[] attrValues)
- void **insertCore** (int layTop, int layBot, boolean matTop, boolean matBot, String material, String topMaterial, String botMaterial, double thickness, double topThickness, double botThickness, String reference, double tolerance, double erConstant, String supplier, ObjectList attrNames, Object[] attrValues, boolean revInsert)
- void **insertCore** (int layTop, int layBot, boolean matTop, boolean matBot, String material, String topMaterial, String botMaterial, double thickness, double topThickness, double botThickness, String reference, double tolerance, double erConstant, String supplier, ObjectList attrNames, Object[] attrValues)
- int **insertDraw** (double line_fx, double line_fy, double line_tx, double line_ty, int iNet, String sSelection)
- int **insertDraw** (**Line** line, int iNet, String sSelection)
- int **insertDraw** (double line_fx, double line_fy, double line_tx, double line_ty)
- int **insertDraw** (**Line** line)
- int **insertFlash** (double pt_x, double pt_y, int iNet, String sSelection)
- int **insertFlash** (**Point** pt, int iNet, String sSelection)
- int **insertFlash** (double pt_x, double pt_y)
- int **insertFlash** (**Point** pt)
- void **insertFlashRepeat** (double pt_x, double pt_y, int iNx, double dXstep, int iNy, double dYstep, String sSelection)
- void **insertFlashRepeat** (**Point** pt, int iNx, double dXstep, int iNy, double dYstep, String sSelection)
- void **insertFlashRepeat** (double pt_x, double pt_y, int iNx, double dXstep, int iNy, double dYstep)
- void **insertFlashRepeat** (**Point** pt, int iNx, double dXstep, int iNy, double dYstep)
- void **insertFullArc3Point** (double pnt1_x, double pnt1_y, double pnt2_x, double pnt2_y, double pnt3_x, double pnt3_y)
- void **insertFullArc3Point** (**Point** pnt1, **Point** pnt2, **Point** pnt3)
- void **insertFullArc3Point** (double pnt1_x, double pnt1_y, double pnt2_x, double pnt2_y, double pnt3_x, double pnt3_y, int iNet, String sSelection)
- void **insertFullArc3Point** (**Point** pnt1, **Point** pnt2, **Point** pnt3, int iNet, String sSelection)
- void **insertFullArcCenterRadius** (double pntCenter_x, double pntCenter_y, double dRadius, String sDirection)
- void **insertFullArcCenterRadius** (**Point** pntCenter, double dRadius, String sDirection)
- void **insertFullArcCenterRadius** (double pntCenter_x, double pntCenter_y, double dRadius, String sDirection, int iNet, String sSelection)
- void **insertFullArcCenterRadius** (**Point** pntCenter, double dRadius, String sDirection, int iNet, String sSelection)
- boolean **insertParallel** (boolean bSelection, ObjectList oLines)

void **insertPolydrawRect** (double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax, boolean bSel)
- void **insertPolydrawRect** (**Rectangle** rect, boolean bSel)
- void **insertPolydrawRect** (double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax)
- void **insertPolydrawRect** (**Rectangle** rect)
- void **insertPolydrawRect** (double drawRectangle_xmin, double drawRectangle_ymin, double drawRectangle_xmax, double drawRectangle_ymax, boolean bRectCW, boolean bSel)
- void **insertPolydrawRect** (**Rectangle** drawRectangle, boolean bRectCW, boolean bSel)
- void **insertPolydrawRect** (**Point** p1, **Point** p2, boolean bRectCW, boolean bSel)
- void **insertPolygon** (boolean bSelection, ObjectList polygon)
- void **insertPrePreg** (int topLayer, int bottomLayer, String sPosition, String material, double thickness, String reference, double tolerance, double erConstant, String supplier, ObjectList attrNames, Object[] attrValues)
- void **insertTab** (double p_x, double p_y, double dis, String pat)
- void **insertTab** (**Point** p, double dis, String pat)
- void **insertVectorText** (double pt_x, double pt_y, String sText, String sFont, double dWidth, double dSpacing, String sMirror, double dRotation, double dScale)
- void **insertVectorText** (**Point** pt, String sText, String sFont, double dWidth, double dSpacing, String sMirror, double dRotation, double dScale)
- void **insertVectorText** (double pt_x, double pt_y, String sText, String sFont, double dWidth, double dSpacing, String sMirror, double dRotation, double dScaleX, double dScaleY)
- void **insertVectorText** (**Point** pt, String sText, String sFont, double dWidth, double dSpacing, String sMirror, double dRotation, double dScaleX, double dScaleY)
- void **intersectDraws** (double pt_x, double pt_y)
- void **intersectDraws** (**Point** pt)
- boolean **isDirectory** (ObjectList fileInfo)
- boolean **isEqual** (Object oParam1, Object oParam2)
- boolean **isFile** (ObjectList fileInfo)
- boolean **isHidden** (ObjectList fileInfo)
- boolean **isLayerInPlane** (int iPlaneNumber)
- void **job_save_shm_and_release** (String sShmName)
- int **jobApeMaxNumber** ()
- String **jobATEMachine** ()
- void **jobAttribute** (String name, String value)
- String **jobAttribute** (String name)
- String **jobAttribute** ()
- void **jobCopperCount** (boolean bUseMask, boolean bConfirmMaskUsage)
- void **jobCopperCount** ()
- void **jobCustomer** (String sCustomer)
- String **jobCustomer** ()
- void **jobDRCParameters** (String sDrc)
- String **jobDRCParameters** ()
- **Rectangle jobEnclosingBox** ()
- void **jobExtension** (String sExtension)
- void **jobFixture** (String sFixture)
- String **jobFixture** ()
- boolean **jobHasPattern** (boolean bUsed)
- void **jobInfo** (String[] sInfo)
- void **jobInfo** (String sInfo)
- String **jobInfo** ()
- void **jobLayMask** (String sLayMask)
- String **jobLayMask** ()
- int **jobMaxNetnumer** ()
- void **jobName** (String sName)
- String **jobName** ()
- boolean **jobNetlist** ()
- int **jobNumApes** ()
- int **jobNumBothExtras** ()
- int **jobNumBothExtras** (String subClass)
- int **jobNumBottomExtras** ()
- int **jobNumBottomExtras** (String subClass)

int **jobNumCores** ()
- int **jobNumDrills** ()
- int **jobNumDrills** (String subClass)
- int **jobNumExtras** ()
- int **jobNumExtras** (String subClass)
- int **jobNumLayers** ()
- int **jobNumNoneExtras** ()
- int **jobNumNoneExtras** (String subClass)
- int **jobNumPrepregs** (int start)
- int **jobNumPrepregs** ()
- int **jobNumSignals** ()
- int **jobNumSignals** (String subClass)
- int **jobNumTopExtras** ()
- int **jobNumTopExtras** (String subClass)
- void **jobPath** (String sPath)
- String **jobPath** ()
- void **jobRevision** (String sRevision)
- String **jobRevision** ()
- int **jobSelectCount** (String sOption)
- int **jobSelectCount** ()
- boolean **jobSelection** ()
- **Rectangle jobSelectionEnclosingBox** ()
- void **jobSize** (double pntSize_x, double pntSize_y)
- void **jobSize** (**Point** pntSize)
- void **jobSize** (String sUnit, double pntSize_x, double pntSize_y)
- void **jobSize** (String sUnit, **Point** pntSize)
- void **jobSize** (String sSize)
- **Point jobSize** ()
- void **jobSpec** (String sSpec)
- String **jobSpec** ()
- void **jobUserData** (String sUserData)
- String **jobUserData** ()
- void **lajCleanLegendLayer** (boolean DoMask, double MaskClearance, boolean DoCu, double CuClearance, boolean DoCuPads, double CuPadClearance, boolean bDoCuFOM, double iCuFOMClearance, boolean bDoCuPadsFOM, double iCuPadsFOMClearance, boolean doPlatedDrills, double platedDrillClearance, boolean doUnplatedDrills, double UnplatedDrillClearance, boolean DoSmallDraws, double MinDrawSize)
- void **lajDefineWord** ()
- void **lajDeselectAllWords** ()
- void **lajDragWord** (double pt_x, double pt_y, double radius, double offset_x, double offset_y, double limit, boolean enforcelimit)
- void **lajDragWord** (**Point** pt, double radius, **Point** offset, double limit, boolean enforcelimit)
- void **lajLegendDRC** (boolean bDoLineWidth, double dMinLineWidth, boolean bDoMask, double dMaskClearance, boolean bDoCu, double dCuClearance, boolean bDoCuPads, double dCuPadClearance, boolean bDoCuFOM, double dCuFOMClearance, boolean bDoCuPadsFOM, double dCuPadsFOMClearance, boolean bDoPlatedDrills, double dPlatedDrillClearance, boolean bDoUnplatedDrills, double dUnplatedDrillClearance, boolean bDoSmallDraws, double dMinDrawSize)
- void **lajLegendTextToWords** (double dMaxSize, int iMaxSpacing)
- void **lajMoveWord** (String value, double dx, double dy, double limit)
- void **lajMoveWord** (String value, double dx, double dy)
- void **lajScaleWord** (String value, double factor, double limit)
- void **lajScaleWord** (String value, double factor)
- void **lajScaleWordOnPt** (double pt_x, double pt_y, double radius, double scale, double limit, boolean enforcelimit)
- void **lajScaleWordOnPt** (**Point** pt, double radius, double scale, double limit, boolean enforcelimit)
- void **lajSelectAllWords** ()
- void **lajUndefineWord** ()
- void **layActive** (ObjectList layerID, boolean bActive)
- boolean **layActive** (ObjectList layerID)
- void **layActive** (boolean bActive)
- boolean **layActive** ()

void **layAlias** (String sAlias)
- String **layAlias** ()
- int **layApeCount** ()
- void **layAttach** (String sAttach)
- String **layAttach** ()
- void **layAttribute** (String name, String value)
- String **layAttribute** (String name)
- String **layAttribute** ()
- void **layClass** (String sNewClass)
- String **layClass** ()
- void **layCopperCount** (boolean bUseMask, boolean bConfirmMaskUsage)
- void **layCopperCount** ()
- **Rectangle layEnclosingBox** ()
- void **layerViewSplit** (boolean on)
- int **layExtractPlotStamps** (String dstLayName, String sOptions, ObjectList sFilters)
- void **layFrom** (int layFrom)
- int **layFrom** ()
- boolean **layHasPattern** (boolean bUsed)
- ObjectList **layID** ()
- void **layIndex** (int iIndex)
- int **layIndex** ()
- void **layInfo** (String sText)
- String **layInfo** ()
- void **layMaterial** (String sMaterial)
- String **layMaterial** ()
- void **layName** (String sName)
- String **layName** ()
- void **layNumber** (int iNumber)
- int **layNumber** ()
- void **layReadable** (String sSide)
- String **layReadable** ()
- void **layReverse** (boolean bReverse)
- boolean **layReverse** ()
- boolean **laySelection** ()
- **Rectangle laySelectionEnclosingBox** ()
- void **laySubClass** (String sSubClass)
- String **laySubClass** ()
- void **layThickness** (double dThickness)
- double **layThickness** ()
- void **layTo** (int layTo)
- int **layTo** ()
- double **layZPos** ()
- void **liftUpUpcbBlocks** ()
- **Line Line** (double ptFromX, double ptFromY, double ptToX, double ptToY, String units)
- **Line Line** (double ptFromX, double ptFromY, double ptToX, double ptToY)
- **Line Line** (**Line** line)
- **Line Line** (**Point** ptFrom, **Point** ptTo)
- ObjectList **listFrames** ()
- void **loadApertures** (String sDpfFile)
- void **loadBuildup** (String buildupSpec)
- void **loadFrames** (boolean bVerbose, boolean bLoadOnce)
- void **loadFrames** ()
- void **loadSplitConfig** (String sConfigName)
- void **loadUFD** (String sUFDName)
- void **loadWorkspace** (String sWorkspaceName)
- void **loadWorkspace** ()
- double **maxInvalidArcsDeviation** ()
- int **measureFingers** (String szOption)
- void **measureLayers** ()
- void **measureObjects** (double p1_x, double p1_y, double p2_x, double p2_y)
- void **measureObjects** (**Point** p1, **Point** p2)

void **measurePoints** (double pt_x, double pt_y)
- void **measurePoints** (**Point** pt)
- void **measurePoints** (double p1_x, double p1_y, double p2_x, double p2_y)
- void **measurePoints** (**Point** p1, **Point** p2)
- void **mergeContours** ()
- void **mergeContoursSingle** ()
- void **mergeContoursSingleAdd** ()
- void **mergeLayers** (String posNegAlt, boolean delLay)
- void **mirror** (String axis, boolean bUseCenter, boolean bOnRefPoints)
- void **models** (String sModelShape, double dTolerance)
- void **models** (String sModelShape)
- boolean **modelsCreateComplex** ()
- boolean **modelsCreateStandard** (double dTolerance)
- **Rectangle modelsDefineSelections** ()
- int **modelsReplace** (double pntTolerance_x, double pntTolerance_y)
- int **modelsReplace** (**Point** pntTolerance)
- int **modelsSelect** (double pntTolerance_x, double pntTolerance_y)
- int **modelsSelect** (**Point** pntTolerance)
- void **modifyCore** (int iTopLay, String sAtt, int iNewTopLay, int iNewBotLay, String sNewAtt, double dThickness, String sMaterial, String sInfo)
- void **modifyDrill** (String sName, String sAlias, String sClass, String sSubClass, int iFrom, int iTo, double dThickness)
- void **modifyExtra** (String sName, String sAlias, String sClass, String sSubClass, String sAttach, int iNumber, boolean bReverse, String sMaterial)
- void **modifyFeedback** (String sName, String sAlias, String sClass, String sSubClass, String sAttach)
- void **modifyLayer** (String sName, String sAlias, String sClass, String sSubClass, int iNumber, boolean bReverse, double dZPosition, String sReadable, String sMaterial, double dThickness)
- void **modifyPrePreg** (int iTopLay, int iIndex, int iNewTopLay, int iNewBotLay, int iNewIndex, double dThickness, String sMaterial, String sInfo)
- void **move** (double pt_x, double pt_y, boolean bOnRefPoints)
- void **move** (**Point** pt, boolean bOnRefPoints)
- void **netlistBuild** (String target)
- void **netlistClear** ()
- void **netlistReference** (String target)
- void **newJob** (String jobPath, String jobName)
- void **notImplemented** (String sFuncName)
- void **objAttribute** (String name, String value)
- String **objAttribute** (String name)
- String **objAttribute** ()
- **Point objCenterPoint** ()
- double **objClearance** ()
- **Rectangle objEnclosingBox** ()
- **Point objFlash** ()
- **Point objFromPoint** ()
- String **objInfo** ()
- int **objNet** ()
- **Point objPoint** ()
- double **objRing** ()
- void **objSelect** (String sel)
- boolean **objSelect** ()
- String **objSense** ()
- String **objShape** ()
- void **objString** (String vtxString)
- String **objString** ()
- **Point objToPoint** ()
- String **objType** ()
- void **offset** (double offset_x, double offset_y)
- void **offset** (**Point** offset)
- **Point offset** ()
- void **offsetX** (double offsetX)
- void **offsetY** (double offsetY)

void **openAboutUcamco** ()
- void **openAboutUcamX** ()
- void **openAdvantools** ()
- void **openAMLIJobManager** ()
- void **openAnamorphicScale** ()
- void **openApeCreator** ()
- void **openApeEditor** ()
- void **openApertureAttributes** ()
- void **openApertureManager** ()
- void **openAttributeEditor** ()
- void **openAttributeManager** ()
- void **openAutoDrill** ()
- void **openAutoDrillEditor** ()
- void **openAutoFixture** ()
- void **openBarcode** ()
- void **openBarcode128** ()
- void **openBoardAnalyzer** ()
- void **openBoardSnapshot** ()
- void **openCalculatorSetup** ()
- void **openCamtek** (String sMachineCfg)
- void **openCFMEEOutput** ()
- void **openCheckList** ()
- void **openCheckListDefineChecklist** ()
- void **openCheckListDefineSteps** ()
- void **openClipping** ()
- void **openColor** ()
- void **openConnect** ()
- void **openContourHandling** ()
- void **openConvertAttributes** ()
- void **openCopperBalance** ()
- void **openCopperRepair** ()
- void **openCoverlayOptimizer** ()
- void **openCU9000Dialog** ()
- void **openDatums** ()
- void **openDistort** ()
- void **openDraw** (double pt_x, double pt_y, double dis)
- void **openDraw** (**Point** pt, double dis)
- void **openDrawSlots** ()
- void **openDRC** ()
- void **openDrillInfo** ()
- void **openDrillMap** ()
- void **openDrillOptimizer** ()
- void **openDrillRoutSetups** ()
- void **openDrillTolerance** ()
- void **openDrillToolManager** ()
- void **openDsAoi** ()
- void **openDsAoiAdvanced** ()
- void **openDSAOIialog** ()
- void **openDsAoiPreview** ()
- void **openDsAoiQueue** ()
- void **openEditingToolbox** ()
- void **openEditVectorText** (double pickPoint_x, double pickPoint_y)
- void **openEditVectorText** (**Point** pickPoint)
- void **openErrors** ()
- void **openEtchCompensation** ()
- void **openExpand** ()
- void **openExternalLinkManager** ()
- void **openFiducials** ()
- void **openFillAngledPattern** ()
- void **openFillPattern** ()
- void **openFillVector** ()

void **openFlashMaker** ()
- void **openFlexManager** ()
- void **openFlipJob** ()
- void **openFrame** (String sFrameName)
- void **openGridParameters** ()
- void **openHelp** ()
- void **openHelpOnHelp** ()
- void **openHelpOnHypertool** ()
- void **openHelpOnResources** ()
- void **openHelpOnVersion** ()
- void **openHiPot** ()
- void **openImageCompare** ()
- void **openImpedanceControl** ()
- void **openImportIPC356** ()
- void **openImportMET** ()
- void **openImportODBxx** ()
- void **openImportWF** ()
- void **openInsertContourText** ()
- void **openInsertVectorText** ()
- void **openJob** (String jobName)
- void **openJob_shm** (String sShmName)
- void **openJobCreate** ()
- void **openJobDefinition** ()
- void **openJobEdit** ()
- void **openJobEditor** ()
- void **openJobEditorOptions** ()
- void **openJobLoad** ()
- void **openJobMerge** ()
- void **openJobPlaneSetup** ()
- void **openJobPrint** ()
- void **openJobView** ()
- void **openLayerEdit** ()
- void **openLegendOptimizer** ()
- void **openLicenseHelp** ()
- void **openLoadCheckList** ()
- void **openMagnifier** ()
- void **openMarkupAssistant** ()
- void **openMessages** ()
- void **openMLIOutput** ()
- void **openModels** ()
- void **openNetCompare** ()
- void **openNetfixSetup** ()
- void **openNonFunctionalPad** ()
- void **openNumbers** ()
- void **openObjectAttributes** ()
- void **openObjectCompare** ()
- void **openOutputAccumatch** ()
- void **openOutputAOI** ()
- void **openOutputCAD** ()
- void **openOutputCamtek** ()
- void **openOutputDrillRout** (String sDrillMachine)
- void **openOutputDsDi** ()
- void **openOutputDsDiPreview** ()
- void **openOutputDsDiQueue** ()
- void **openOutputNetlist** ()
- void **openOutputOrbot** ()
- void **openOutputSapphire** ()
- void **openOutputScoring** ()
- void **openOutputSmartArgos** ()
- void **openOutputTrackscan** ()
- void **openOutputUxpAutomanager** ()

void **openOutputUxpEtec** ()
- void **openOutputUxpLocal** ()
- void **openPanelFramesCoupons** ()
- void **openPanelLinks** ()
- void **openPanelPlus** ()
- void **openPanelReproduce** ()
- void **openPanelSetup** ()
- void **openPanelStepRepeat** ()
- void **openPlaneAdjuster** ()
- void **openPlotParameters** ()
- void **openPPMonitor** ()
- void **openProductionStagesEditor** ()
- void **openQueryNet** ()
- void **openQueryObject** ()
- void **openReferencePoints** ()
- void **openRegister** ()
- void **openRemoveAttributes** ()
- void **openRepair** (String szLabname)
- void **openRoutManager** ()
- void **openRoutManagerCleanUp** ()
- void **openRoutManagerDimensioning** ()
- void **openRoutManagerEditor** ()
- void **openRoutManagerTools** ()
- void **openSaveJobAs** ()
- void **openSaveLayout** ()
- void **openSecureEtchCompensation** ()
- void **openSelections** ()
- void **openSetupOptions** ()
- void **openSetupSave** ()
- void **openShavePads** ()
- void **openSignalLayerAdjuster** ()
- void **openSignalLayerAdjusterAssistant** ()
- void **openSilkOptimizer** ()
- void **openSmartCamtek** (String sMachineCfg)
- void **openSmartDRC** ()
- void **openSmartFix** ()
- void **openSmartplot** ()
- void **openSmartSR** ()
- void **openSmartStart** ()
- void **openSoldermask** ()
- void **openSoldermaskOptimizer** ()
- void **openTearDrop** ()
- void **openTechnicalAnalyzer** ()
- void **openTestpointEdit** ()
- void **openToolbarManager** ()
- void **openToolbars** ()
- void **openTransformObjects** ()
- void **openTransformObjectsBGAPads** ()
- void **openTransformObjectsBGATracks** ()
- void **openTransformObjectsEdit** ()
- void **openTransformObjectsRescale** ()
- void **openUcamDbEditor** ()
- void **openUndoRedoDetails** ()
- void **openUTest** ()
- void **openUtestUtilities** ()
- void **openValidateLayer** ()
- void **openVectorTextFont** ()
- void **openVerifyArcsDraws** ()
- void **openViewGuide** ()
- void **openYsphotechOutput** ()
- void **optimizeDrill** (int nPasses, int optMode, double yxTime)

void **optimizeMaskLayer** (double dMinRing, double dMaxRing, double dMaskToCopper, double dMaskToMask, double dBigRing)
- String **osChDir** ()
- String **osChDir** (String sDir)
- int **osCopy** (String sSrcName, String sDstName)
- String **osCreateTmpDir** (String sBasePath)
- String **osCreateTmpDir** ()
- int **osDelete** (String sFileName)
- ObjectList **osFileInfo** (String sPath)
- ObjectList **osGetFileList** (String sDir, String sFileMask, boolean bRecurse, boolean bFullPath, boolean bWithDirs)
- ObjectList **osGetFileList** (String sDir, String sFileMask, boolean bRecurse, boolean bFullPath)
- ObjectList **osGetFileList** (String sDir, String sFileMask, boolean bRecurse)
- ObjectList **osGetFileList** (String sDir, String sFileMask)
- ObjectList **osGetFileList** (String sDir, boolean bRecurse)
- ObjectList **osGetFileList** (String sDir, boolean bRecurse, boolean bFullPath, boolean bWithDirs)
- ObjectList **osGetFileList** (String sDir)
- int **osMarkAsTmp** (String sName)
- int **osMkDir** (String sDirName)
- int **osMove** (String sSrcName, String sDstName)
- void **osRmDir** (String sDirName)
- void **osRmTree** (String sDirName)
- int **osUnTgz** (String sTgzArchive, String sDstDir)
- int **osUnZip** (String sZipArchive, String sDstDir)
- void **outAtgFixture** (String key, String sTool, String iRes, int iSession)
- void **output274x** (String sRes)
- void **outputAft** (String res)
- void **outputAoi** (boolean bCadData, boolean bReference)
- void **outputAtf** ()
- ObjectList **outputAutoDrill** (String sDrjFile)
- void **outputCFMEE** (String outputPath, boolean reverse, double marginx, double marginy, boolean distort, double distortx, double distorty, double resizex, double resizey, boolean deleteOutside)
- void **outputCli** ()
- void **outputColorPDF** (String sPdfFullPath)
- void **outputDp40** (double pt_x, double pt_y, boolean bPositive, boolean bMirrorx, boolean bMirrory, double dLaserPower, int iPolygonSpeed, int iPcbFormat, String unit)
- void **outputDp40** (**Point** pt, boolean bPositive, boolean bMirrorx, boolean bMirrory, double dLaserPower, int iPolygonSpeed, int iPcbFormat, String unit)
- void **outputDxf** (String unit, int iConturize, int iKeepTXT, double dExpandArcs, int iCenterLine, int iAllInOne)
- void **outputDxfV6** (String unit)
- void **outputEie** (String sRes, ObjectList par)
- void **outputEtec** (String sResistOuter, String sResistInner, double mediaX, double mediaY, int iAlignType, int iLevelType, int iCycles, String sDate, String sTime, boolean bAuto, double dScaleX, double dScaleY, double dScaleOriX, double dScaleOriY, String sMDFfile, String sResource)
- void **outputExt** (String lan, String too, String res, ObjectList resdb, String inc1, String inc2, int session, Object[] pre, Object[] pos, Object notUsed)
- void **outputExt** (String lan, String too, String res, ObjectList resdb, String inc1, String inc2, int session, Object[] pre, Object[] pos)
- void **outputHimt** (double datum_x, double datum_y, double offset_x, double offset_y, String mirror, String rotation)
- void **outputHimt** (**Point** datum, **Point** offset, String mirror, String rotation)
- void **outputIpc2581** ()
- void **outputIpcUfd** (String key, String res, String version)
- void **outputLpg** (int iPpi, int iChoke, double offset_x, double offset_y)
- void **outputLpg** (int iPpi, int iChoke, **Point** offset)
- int **outputManiaSapphire** (String sOutputPath, String sDescription, String sGeometryfile, boolean bStatistics, boolean bDrill)
- void **outputMda** (String sPath, ObjectList par, Object[] subpar, int iApr, int iSubfig, int iRenum)
- void **outputNec** (String too)
- void **outputOdbxx** (String res)

void **outputOdbxxv7** (String res)
- void **outputOif** (String oifVersion, int byJob, int pan, int fillin)
- boolean **outputOrbot** ()
- void **outputPdf** ()
- void **outputProbe** (String sLang, int iSession, int iAccuracy)
- int **outputRaid** ()
- void **outputRpd** (int iPpi, double datum_x, double datum_y, double offset_x, double offset_y, String sMirror, String sRotation)
- void **outputRpd** (int iPpi, **Point** datum, **Point** offset, String sMirror, String sRotation)
- boolean **outputSchmid** (int resolution, double maskRectangle_xmin, double maskRectangle_ymin, double maskRectangle_xmax, double maskRectangle_ymax, int maskRotation, String maskMirror, String maskPolarity, int equipmentRotation, String equipmentMirror, double offsetX, double offsetY, ObjectList fiducials, String imagePath, String configPath, String batchFile)
- boolean **outputSchmid** (int resolution, **Rectangle** maskRectangle, int maskRotation, String maskMirror, String maskPolarity, int equipmentRotation, String equipmentMirror, double offsetX, double offsetY, ObjectList fiducials, String imagePath, String configPath, String batchFile)
- void **outputSl13** (String sResources, String sKey)
- void **outputSprint** (String sOutputFolder, boolean bStandardMark, String sCopperName, String sRefName, String sAttributeZero, int iZeroPointNumber, String sAttributeCamera, int iCameraNumber, int iRotation0, int iRotation90, int iRotation180, int iRotation270, String sText1, String sText2, String sText3, String sCleanOption)
- void **outputSys** ()
- void **outputTiff** (String sPath, String sExt, String sOptions, int iResolution)
- int **outputTs3** (boolean bDrilledBoards, double dSpace)
- void **outputWf2** (ObjectList par)
- void **outputXdpf** (String sPath)
- void **outputYsphotech** (boolean imagecomp, String outputPath, String reverse, double marginx, double marginy, boolean mirrorx, boolean mirrory, double rotate, double distortx, double distorty, double resize, boolean keepArrays, boolean deleteOutside, boolean autoDetected, String layername)
- void **outputYsphotech** (String outputPath, String reverse, double marginx, double marginy, boolean mirrorx, boolean mirrory, double rotate, double distortx, double distorty, double resize, boolean keepArrays, boolean deleteOutside, boolean autoDetected, String layername)
- void **pajPlaneAdjust** (double dPlatedClearance, double dUnplatedClearance, double dRingSize, double dRingClearance, double dLineWidth, double dCuClearance, boolean bDoCut, double dOutlineCLearance)
- void **pajPlaneAdjust** (double dPlatedClearance, double dUnplatedClearance, double dRingSize, double dRingClearance, double dLineWidth, double dCuClearance, boolean bDoCut, double dOutlineClearance, boolean bOutputAsContour, boolean bSaveBackup, boolean bErrorPopups)
- void **pajPlaneAdjust** (double dPlatedClearance, double dUnplatedClearance, double dRingSize, double dRingClearance, double dLineWidth, double dCuClearance, boolean bDoCut, double dOutlineClearance, boolean bOutputAsContour, boolean bSaveBackup)
- void **panelStepRepeat** (double pStart_x, double pStart_y, int iRepeatX, int iRepeatY, double dStepX, double dStepY, String sFlashPoint)
- void **panelStepRepeat** (**Point** pStart, int iRepeatX, int iRepeatY, double dStepX, double dStepY, String sFlashPoint)
- void **panelStepRepeatCenter** (double pStart_x, double pStart_y, int iRepeatX, int iRepeatY, double dStepX, double dStepY)
- void **panelStepRepeatCenter** (**Point** pStart, int iRepeatX, int iRepeatY, double dStepX, double dStepY)
- void **panelStepRepeatJobZero** (double pStart_x, double pStart_y, int iRepeatX, int iRepeatY, double dStepX, double dStepY)
- void **panelStepRepeatJobZero** (**Point** pStart, int iRepeatX, int iRepeatY, double dStepX, double dStepY)
- void **panelStepRepeatMiddle** (double pStart_x, double pStart_y, int iRepeatX, int iRepeatY, double dStepX, double dStepY)
- void **panelStepRepeatMiddle** (**Point** pStart, int iRepeatX, int iRepeatY, double dStepX, double dStepY)
- int **panelStepRepeatValidate** ()
- void **pasteFromClipboard** ()
- String **peGetInputFile** ()
- boolean **peGetInputJobBooleanProperty** (String name)
- double **peGetInputJobDoubleProperty** (String name)
- int **peGetInputJobIntegerProperty** (String name)
- String **peGetInputJobProperty** (String name)
- String **peGetJobList** ()

int **peGetOptionalQuantity** ()
- boolean **peGetPanelJobBooleanProperty** (String name)
- double **peGetPanelJobDoubleProperty** (String name)
- int **peGetPanelJobIntegerProperty** (String name)
- String **peGetPanelJobProperty** (String name)
- int **peGetPCBQuantity** ()
- boolean **peGetSingleOptimization** ()
- boolean **peGetSolutionBooleanProperty** (String name)
- double **peGetSolutionDoubleProperty** (String name)
- int **peGetSolutionIntegerProperty** (String name)
- String **peGetSolutionProperty** (String name)
- boolean **peGetUseFrameSet** ()
- void **peSetInputFile** (String fileName)
- void **peSetInputJobProperty** (String name, boolean value)
- void **peSetInputJobProperty** (String name, double value)
- void **peSetInputJobProperty** (String name, int value)
- void **peSetInputJobProperty** (String name, String value)
- void **peSetOptionalQuantity** (int quantity)
- void **peSetPanelJobProperty** (String name, boolean value)
- void **peSetPanelJobProperty** (String name, double value)
- void **peSetPanelJobProperty** (String name, int value)
- void **peSetPanelJobProperty** (String name, String value)
- void **peSetSingleOptimization** (boolean set)
- void **peSetSolutionProperty** (String name, boolean value)
- void **peSetSolutionProperty** (String name, double value)
- void **peSetSolutionProperty** (String name, int value)
- void **peSetSolutionProperty** (String name, String value)
- void **peSetUseFrameSet** (boolean set)
- void **pickAperture** (double pt_x, double pt_y, double radius)
- void **pickAperture** (**Point** pt, double radius)
- **Point pickPoint** (String sLabel)
- void **plotAddLayerToMerge** (String sJobName, String sPath)
- void **plotAddLayerToMerge** ()
- boolean **plotLayer** (String sPath, int iFillPercentage, boolean bSeparator, boolean bClean)
- boolean **plotLayer** (int iFillPercentage, boolean bSeparator, boolean bClean)
- boolean **plotMergedLayers** (int iFillPercentage, boolean bSeparator, boolean bClean)
- void **plotResetParams** ()
- void **plotSetAttribute** (ObjectList oLayID, String sName, String sValue)
- void **plotSetAttribute** (String sName, String sValue)
- void **plotSetAttribute** (String sName)
- void **plotSetParam** (ObjectList oLayID, String sKey, String sName, double dValue)
- void **plotSetParam** (String sKey, String sName, double dValue)
- void **plotSetParam** (ObjectList oLayID, String sKey, boolean bValue)
- void **plotSetParam** (String sKey, boolean bValue)
- void **plotSetParam** (ObjectList oLayID, String sKey, int iValue)
- void **plotSetParam** (String sKey, int iValue)
- void **plotSetParam** (ObjectList oLayID, String sKey, double dValue)
- void **plotSetParam** (String sKey, double dValue)
- void **plotSetParam** (ObjectList oLayID, String sKey, String sValue)
- void **plotSetParam** (String sKey, String sValue)
- void **plotSetRipHost** (String sRIP)
- void **plotStartNew** ()
- **Point Point** (**Point** point)
- **Point Point** (double x, double y, String units)
- **Point Point** (double x, double y)
- void **point1** (double point1_x, double point1_y)
- void **point1** (**Point** point1)
- **Point point1** ()
- void **point1Active** (boolean bActivate)
- boolean **point1Active** ()
- void **point1X** (double pt1X)

void **point1Y** (double pt1Y)
- void **point2** (double point2_x, double point2_y)
- void **point2** (**Point** point2)
- **Point point2** ()
- void **point2Active** (boolean bActivate)
- boolean **point2Active** ()
- void **point2X** (double pt2X)
- void **point2Y** (double pt2Y)
- void **printListRefPoints** (boolean bOnAllActiveLay)
- void **printListRefPoints** ()
- boolean **promptBoolean** (String optName, boolean def)
- double **promptDouble** (String doubleName, double def)
- void **promptEnd** ()
- String **promptFileName** (String strLabel, String def)
- int **promptInteger** (String intName, int def)
- void **promptLabel** (String labelText)
- **Line promptLine** (String lineName, double fromX, double fromY, double toX, double toY)
- **Line promptLine** (String lineName, **Line** defLine)
- String **promptOption** (String optName, ObjectList options, String def)
- **Point promptPoint** (String pointName, double ptX, double ptY)
- **Point promptPoint** (String pointName, **Point** point)
- **Rectangle promptRectangle** (String rectangleName, double xmin, double xmax, double ymin, double ymax)
- **Rectangle promptRectangle** (String rectangleName, **Rectangle** rectangle)
- void **promptStart** (String sSetName, String sTitle)
- void **promptStart** (String sSetName)
- void **promptStart** ()
- String **promptString** (String strName, String def)
- double **promptUnit** (String unitName, double def, String units)
- void **qmerge** (String sOptions)
- void **quitBlockEdit** (boolean bSave, boolean bKeepLink)
- void **quitBlockEdit** (boolean bSave)
- void **quitBlockMultiEdit** (boolean bSave, boolean bKeepLink)
- void **quitBlockMultiEdit** (boolean bSave)
- void **quitComplexEdit** (boolean bSave)
- void **quitConfirm** ()
- void **readAmli** (String sPath)
- void **recognizeContours** (String sConName)
- **Rectangle Rectangle** (**Rectangle** rect)
- **Rectangle Rectangle** (double xmin, double ymin, double xmax, double ymax, String units)
- **Rectangle Rectangle** (double xmin, double ymin, double xmax, double ymax)
- **Rectangle Rectangle** (**Point** ptPoint1, **Point** ptPoint2)
- void **redo** ()
- void **registerJobOnPoints** ()
- void **registerLayers** ()
- void **registerOnGrid** (double GridStep_x, double GridStep_y, double GridOri_x, double GridOri_y, double dXRadius, double dYRadius)
- void **registerOnGrid** (**Point** GridStep, **Point** GridOri, double dXRadius, double dYRadius)
- void **registerOnPads** (double dRadius, boolean bOnFlashPoint)
- void **removeApeAttr** ()
- void **removeJobAttr** ()
- void **removeLayAttr** ()
- void **removeNetAttr** (int iNetNumber, String sAttrName, String sAttrValue)
- void **removeNetAttr** (int iNetNumber, String sAttrName)
- void **removeNetAttr** (String sAttrName, String sNetName)
- void **removeNetAttr** (String sAttrName)
- void **removeObjAttr** ()
- void **removeObjectAttribute** (String sAttrName, String sAttrValue)
- void **removeObjectAttribute** (String sAttrName)
- void **removeYsphotechPlotstamp** (int plotstampID)
- void **replaceApeByCurrent** ()

void **replaceApertures** ()
- void **replaceBitmapByContours** ()
- void **replaceDrawsWithArcs** (double tolerance)
- void **replaceInnersByOuters** ()
- void **replaceZeroLengthDraws** (double dMaxLength, boolean bFunctional, boolean bNonFunctional)
- void **reproducePanel** (String report)
- void **reset** ()
- void **resetCFMEE** ()
- void **resetCores** (int iTop, String sAttach)
- void **resetCores** ()
- void **resetWorkspace** ()
- void **resetYsphotech** ()
- void **restoreArcs** (boolean bPreferFullArc)
- void **restoreContours** (boolean bPreferFullArc)
- void **returnVariables** (ObjectList returnVariables)
- void **reverse** ()
- void **reverseLayer** ()
- void **reverseLayers** ()
- void **rotate** (double angle, boolean bUseCenter, boolean bOnRefPoints)
- void **roundDraw** (double pt_x, double pt_y, double dis)
- void **roundDraw** (**Point** pt, double dis)
- void **routStatistics** (String doOption)
- void **routStatistics** ()
- void **runDRC** (String sCfgFile, boolean bBuildNetlist, String sUseNetlist, boolean bSelErrors)
- String **runFile** (String sScriptPath, ObjectList argv)
- String **runFile** (String sScriptPath)
- String **runScript** (String sScript, ObjectList argv)
- String **runScript** (String sScript)
- ObjectList **runScriptWithReturn** (String sScript, Object[] argv)
- ObjectList **runScriptWithReturn** (String sScript)
- void **saveAmli** (String sPath)
- void **saveBuildup** (String sSpec, String sCustomer, String sDrcPar, String sCoreRef, String sPrePregMat, String sCopperMat, String sJobFlow, String sTechCheck, String sAttrSet, String sDatumList, ObjectList layList)
- int **saveJob** ()
- int **saveJobAs** (String fullPath, String sVersion)
- int **saveJobAs** (String fullPath)
- void **saveJobAsV3** ()
- void **saveJobAsV6** ()
- void **saveJobAsV9** ()
- void **saveLayer** (String sClass, String sSubClass, int iLayIndex, String sFullPath)
- void **saveLayer** (String layName, String fullPath)
- void **saveMessagesAs** (String sFilePath)
- void **saveOrder** ()
- void **saveSplitConfig** (String sConfigName)
- void **saveUFD** (String sUFDName)
- void **saveWorkspace** ()
- void **saveWorkspace** (String sWorkspaceName)
- void **saveWorkspaceAs** (String sWorkspaceName)
- void **scale** (double dScaleValue, boolean bUseCenter)
- void **scaleObjectOnAttribute** (String sName, double dScaleFactor, double dMinClearance)
- void **screendump** ()
- void **secureEtchCompensation** (double dPadSpread, double dSmdSpread, double dTrackSpread, double dAreaSpread, double dPadPadClearance, double dPadSmdClearance, double dPadTrackClearance, double dPadAreaClearance, double dSmdSmdClearance, double dSmdTrackClearance, double dSmdAreaClearance, double dTrackTrackClearance, double dTrackAreaClearance, double dAreaAreaClearance, String sContourMethod, boolean bProcessSameNetSpacing, boolean bBackupOriginalLayer, boolean bCheckMissingCopper, boolean bFastMode, int iShiftMode, double dMinCopper)
- void **secureEtchCompensation** (double dPadSpread, double dSmdSpread, double dTrackSpread, double dAreaSpread, double dPadPadClearance, double dPadSmdClearance, double dPadTrackClearance,

double dPadAreaClearance, double dSmdSmdClearance, double dSmdTrackClearance, double dSmdAreaClearance, double dTrackTrackClearance, double dTrackAreaClearance, double dAreaAreaClearance, String sContourMethod, boolean bProcessSameNetSpacing, boolean bBackupOriginalLayer, boolean bCheckMissingCopper, boolean bFastMode, int iShiftMode)
- void **secureEtchCompensation** (double dPadSpread, double dSmdSpread, double dTrackSpread, double dAreaSpread, double dPadPadClearance, double dPadSmdClearance, double dPadTrackClearance, double dPadAreaClearance, double dSmdSmdClearance, double dSmdTrackClearance, double dSmdAreaClearance, double dTrackTrackClearance, double dTrackAreaClearance, double dAreaAreaClearance, String sContourMethod, boolean bProcessSameNetSpacing, boolean bBackupOriginalLayer, boolean bCheckMissingCopper, boolean bFastMode)
- void **secureEtchCompensationUndo** ()
- void **selectAll** ()
- void **selectAll** (String selectMode)
- void **selectAllApertures** ()
- void **selectAllContours** (String selectMode, String conMode)
- void **selectAllContours** (String selectMode, double xSize, double ySize, String conMode)
- void **selectAmbiguousContours** (String selectMode)
- void **selectAperture** (ObjectList apeIndexArray)
- void **selectAperture** ()
- void **selectAperturesBiggerThan** (String selectMode, double dx, double dy)
- void **selectAperturesSmallerThan** (String selectMode, double dx, double dy)
- void **selectByApeAttributeNames** (String selectMode, String[] sName)
- void **selectByApertureShape** (String selectMode, String apertureShapes)
- void **selectByAttributeName** (String selectMode, String sName)
- void **selectByAttributeValue** (String selectMode, String sName, String sValue)
- void **selectByObjectType** (String selectMode, String objectTypes)
- void **selectChained** (String selectMode, double pt_x, double pt_y)
- void **selectChained** (String selectMode, **Point** pt)
- void **selectChained** (String selectMode, double pt_x, double pt_y, double dTolerance)
- void **selectChained** (String selectMode, **Point** pt, double dTolerance)
- void **selectChainedObjects** (String selectMode, double pnt_x, double pnt_y, double pixelRadius, double dOffCenter, boolean bSameApe, boolean bSameOrientation)
- void **selectChainedObjects** (String selectMode, **Point** pnt, double pixelRadius, double dOffCenter, boolean bSameApe, boolean bSameOrientation)
- void **selectContourByClick** (String selectMode, double pt_x, double pt_y, String conMode)
- void **selectContourByClick** (String selectMode, **Point** pt, String conMode)
- void **selectContoursInWindow** (String selectMode, double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax, String conMode)
- void **selectContoursInWindow** (String selectMode, **Rectangle** rect, String conMode)
- void **selectContoursWithThinRegion** (double dThin, String selectMode)
- void **selectCurrentAperture** (String selectMode)
- void **selectCurrentApertureDefinition** (String selectMode)
- void **selectCurrentObject** (String selectMode)
- void **selectDoubles** (String selectMode, double tolerance)
- void **selectEmbedded** (String selectMode, double tolerance)
- void **selectFlashesLongerThan** (String selectMode, double rRefRatio)
- int **selectHornablePads** (String selectMode, String apertureShapes)
- boolean **selectInvalidArcs** ()
- int **selectInvalidArcs** (String sSelectMode, double dDeviation, String sLimit)
- void **selectIsolatedFlashes** (String selectMode)
- void **selectMesh** (String selectMode)
- int **selectNetByClick** (String selectMode, double pt_x, double pt_y)
- int **selectNetByClick** (String selectMode, **Point** pt)
- int **selectNetByName** (String selectMode, String sNetName)
- void **selectNetByNumber** (String selectMode, int net, boolean bSelectShaved, boolean bSelectBroken)
- void **selectNetByNumber** (String selectMode, int net)
- void **selectNetByTestpoints** (String selectMode, int nbt)
- void **selectNetsWithoutPads** (String selectMode)
- String **selectNonFunctionalPads** ()
- void **selectObjectAttribute** (String sAttrName, String sAttrValue)
- void **selectObjectAttribute** (String sAttrName)

void **selectObjectByAttribute** (String sAttrName, String sAttrValue)
- void **selectObjectByAttribute** (String sAttrName)
- void **selectObjectByAttributeName** (String selectMode, String sName)
- void **selectObjectByAttributeValue** (String selectMode, String sName, String sValue)
- void **selectObjectByShape** (String selectMode, String apertureShapes)
- void **selectObjectByType** (String selectMode, String objectTypes)
- void **selectOpenContours** (String selectMode)
- void **selectOverlappingContours** (String selectMode)
- void **selectOverlaps** (String selectMode)
- void **selectPainted** (String selectMode)
- int **selectPaintedAreas** (boolean bUseLoops, boolean bExcludeChains)
- int **selectPaintedAreas** ()
- void **selectPlotStamps** (String selectMode)
- void **selectPolygon** (boolean bInside, boolean bOutside, boolean bCrossing, String sShapes, String sObjects, String selectMode, ObjectList polygonPoints)
- void **selectReferenceLayer** (String selectMode)
- void **selectReverse** (String selectMode)
- void **selectSmallContours** (String selectMode, double xSize, double ySize, String conMode)
- void **selectSmallSurface** (String selectMode, double surface, String conMode)
- void **selectSmallTracks** (String selectMode, String lenMode, String dstMode, double maxLength)
- void **selectTouchingObjects** (String selectMode, double pnt_x, double pnt_y, double pixelRadius)
- void **selectTouchingObjects** (String selectMode, **Point** pnt, double pixelRadius)
- void **selectWindow** (String selectMode, double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax, String winopt, String sShapes, String sObjects)
- void **selectWindow** (String selectMode, **Rectangle** rect, String winopt, String sShapes, String sObjects)
- void **selectZeroLengthDraws** (double dMaxLength, boolean bFunctional, boolean bNonFunctional)
- boolean **setApe** (int index)
- void **setApertureAttribute** (String sAttributeName, String sAttributeValue)
- void **setAttributeOnObject** (String attrName, String attrValue)
- boolean **setCurrentAperture** (int iIndex)
- void **setInPlane** (int newPlane, int iIndex)
- void **setInPlane** (int newPlane, String layName)
- void **setInPlane** (int newPlane, String layClass, String laySubclass, String attach, int index)
- void **setInPlane** (int newPlane, String layClass, String laySubclass, String attach, int index, boolean activate)
- void **setInPlane** (int newPlane, ObjectList layerID, boolean activate)
- void **setInPlaneByName** (int newPlane, String layName, boolean activate)
- void **setLayerViewBottom** (ObjectList nameArray)
- void **setLayerViewDrill** (ObjectList nameArray)
- void **setLayerViewMain** (ObjectList nameArray)
- void **setLayerViewTop** (ObjectList nameArray)
- void **setMode** (ObjectList oMode)
- void **setMode** (String sParams)
- void **setMode** (String sOption, String sUnit, String sSnapMode)
- Ulayer **setNextLayerToPlane1** ()
- void **setOrigin** (double p_x, double p_y)
- void **setOrigin** (**Point** p)
- void **setOrigin** (double pt_x, double pt_y, boolean bOnRefPoints)
- void **setOrigin** (**Point** pt, boolean bOnRefPoints)
- void **setOriginCenter** (double p_x, double p_y, boolean useOutline)
- void **setOriginCenter** (**Point** p, boolean useOutline)
- void **setOriginToCenter** (boolean bUseOutline, boolean bOnRefPoints)
- void **setPlotParam** (String sKey, int iValue)
- void **setPlotParam** (String sKey, double dValue)
- void **setPlotParam** (String sKey, String sValue)
- void **setResolution** (int resolution)
- void **setSnap** (String sMode)
- void **setSnapOnContour** (boolean on)
- void **setUnit** (String unit)
- void **setYsphotechAlignmentPointType** (int region, int point, String type)
- void **setYsphotechPlotstamp** (int plotstampID, double rec_xmin, double rec_ymin, double rec_xmax,

double rec_ymax, String type)
- void **setYsphotechPlotstamp** (int plotstampID, **Rectangle** rec, String type)
- void **shavePads** (double dPadTraClr, double dPadPadClr, int iClip, boolean bShaveInsideCom)
- void **shavePads** (double dPadTraClr, double dPadPadClr, int iClip)
- void **shavePadsOnMaskLayer** (double dPadToTrack, double dPadToPad)
- void **showBlockStructure** ()
- void **showMeasureValues** (double p1_x, double p1_y, double dx, double dy, double clr, double rng)
- void **showMeasureValues** (**Point** p1, double dx, double dy, double clr, double rng)
- void **showNetlistProfile** ()
- void **silkOptimize** (int iReference, double dClearanceToReference, boolean bCompensateBumps, double dBumpClearance, int iMethod, double dMinimumDrawLength)
- int **smoothen** (String mode, double dMaxDeviation)
- int **smoothen** (String mode, double dMaxDeviation, int iMinReplacePoints)
- void **spawn_func** (String sCommand)
- int **splitContour** (double dOverlapX, double dOverlapY, double dMinX, double dMinY)
- void **splitContours** ()
- boolean **stackupByGerAttr** ()
- void **standardizeBoxes** ()
- void **testVDPathfinder** (int iStart, int iEnd)
- void **testVDPathfinder2** (double dStartX, double dStartY, double dEndX, double dEndY)
- void **toggleApertureSelections** ()
- void **toggleSelections** ()
- void **toggleViewInBlocks** ()
- void **toggleViewMode** ()
- void **toggleViewObjects** ()
- void **toggleViewRefPoints** ()
- void **toggleViewZero** ()
- void **trimDraws** (double p1_x, double p1_y, double p2_x, double p2_y)
- void **trimDraws** (**Point** p1, **Point** p2)
- void **undo** ()
- void **undoClear** ()
- void **unload** (String sClass, String sSubClass, int iLayIndex, boolean bSave)
- void **updateProbes** ()
- void **updateTestPoints** ()
- void **updateTestPointsAndProbes** ()
- void **updateZPosition** ()
- void **utestCheck3DProbeClearance** (boolean bCheck3DProbeClearance, double dClearance)
- void **utestCreateEtmComponentLayers** (boolean bCreateCompLay)
- void **utestDedicatedFixtures** (boolean bDedicatedFixtures, boolean bFirstProbeNumber0, double dFontScale, int iShowProbeNumberEvery, boolean bShowConnectorNumberAlways, boolean bContinousNumbering, boolean bContinousNumberingOnBottom)
- void **utestDo** ()
- void **utestFiducals** (boolean bFiducals)
- void **utestFixtureSizeSplit** (boolean bFixtureSizeSplit, int iSplitSet)
- void **utestGuidePlates** (boolean bGuidePlates)
- void **utestKelvin4WireTest** (boolean bKelvin4WireTest, boolean bUsedMidPoints, boolean bTestOnBlindHoles, boolean bTestOnlyThroughHoles, boolean bTestAllPads, double dMinDrillDia, double dMaxDrillDia, int iSearchDepthLimit)
- void **utestKelvin4WireTest** (boolean bKelvin4WireTest, boolean bUsedMidPoints, boolean bTestOnBlindHoles, boolean bTestOnlyThroughHoles, double dMinDrillDia, double dMaxDrillDia, int iSearchDepthLimit)
- void **utestMachine** (int iSession, String sMachName, String sAccesstype)
- void **utestMicroAdjustment** (boolean bMicroAdjustment, int iNbrOfTestPoints, double dTestPointDiameter, double dTestPointShiftEdge, double dTestPointShiftValue, double dTestPointPitch, double dClearanceFactor, double dCenterDiameter)
- void **utestNetlist** (boolean bNetlist, boolean bNetlistBuild, boolean bNetlistExpand)
- void **utestOutput** (boolean bOutput, boolean bDrillFixture, String sDrillFixture, boolean bNetlist, String sNetlist, boolean bElectTest, boolean bPinInserter, boolean bRepairAid, String sRepairAid)
- void **utestProbeAssignment** (boolean bProbeAssignment, boolean bStagger, String sStagpnt, boolean bStagopttrian, boolean bStagoptlined, double dPitch, double dTolerance, double dSetBack, boolean bReverse, boolean bAxis)

void **utestProbeMapping** (boolean bProbeMapping)
- void **utestTestpoints** (boolean bTestPoints, int iLoop, boolean bUseMasks, boolean bProbeSwaping, boolean bHandlePaintedPads, boolean bCircuitryCheck, boolean bFilterCopperAreas, boolean bViaOfSMDs, boolean bDrillsWithoutPad)
- void **utestTestpointsBOT** (boolean bPointsBot1, boolean bPointsBot2, boolean bPointsBot3, boolean bPointsBot4, boolean bPointsBot5, boolean bPointsBot6, boolean bPointsBot7)
- void **utestTestpointsTOP** (boolean bPointsTop1, boolean bPointsTop2, boolean bPointsTop3, boolean bPointsTop4, boolean bPointsTop5, boolean bPointsTop6, boolean bPointsTop7)
- void **validateInvalidArcs** ()
- void **viewAmbiguous** ()
- void **viewGrid** (boolean bVisible, double ptOrigin_x, double ptOrigin_y, double dXStep, double dYStep, boolean bCross)
- void **viewGrid** (boolean bVisible, **Point** ptOrigin, double dXStep, double dYStep, boolean bCross)
- void **viewGrid** (boolean bVisible, **Point** ptOrigin, **Point** ptStep, boolean bCross)
- void **viewGrid** (boolean bVisible)
- void **viewGrid** ()
- void **viewGuide** ()
- void **viewHistory** ()
- void **viewInBlocks** (boolean trueFalse)
- void **viewMessages** ()
- void **viewMode** (String sMode)
- void **viewModeFilled** ()
- void **viewModeOutline** ()
- void **viewModeSkeleton** ()
- void **viewNumbers** ()
- void **viewObjects** (boolean trueFalse)
- void **viewPan** (double p1_x, double p1_y, double p2_x, double p2_y)
- void **viewPan** (**Point** p1, **Point** p2)
- void **viewRefPoints** (boolean trueFalse)
- void **viewRepaint** ()
- void **viewWarning** (String message)
- void **viewZero** (boolean trueFalse)
- void **viewZoom** (String sZoom)
- void **viewZoomIn** ()
- void **viewZoomOut** ()
- void **viewZoomSelections** ()
- void **viewZoomTotal** ()
- void **viewZoomWindow** ()
- void **viewZoomWindow** (double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax, boolean bScreenCenter)
- void **viewZoomWindow** (**Rectangle** rect, boolean bScreenCenter)
- void **viewZoomWindow** (double rect_xmin, double rect_ymin, double rect_xmax, double rect_ymax)
- void **viewZoomWindow** (**Rectangle** rect)
- void **xmlAdd** (String sDataName, String sElementName, String sContent, ObjectList attrArray)
- void **xmlAdd** (String sDataName, String sElementName, String sContent)
- void **xmlAdd** (String sElementName, String sContent)
- void **xmlAddData** (String sParentDataName, String sDataName)
- void **xmlAddData** (String sDataName)
- void **xmlCreateData** (String sDataName)
- void **xmlDocument** (String rootName)
- void **xmlSave** (String sDestFilePath)
- void **YachiyoAOI_clearOutput** (String name)
- void **YachiyoAOI_defineArea** (int grpIndex, int areaIndex, double pos_x, double pos_y)
- void **YachiyoAOI_defineArea** (int grpIndex, int areaIndex, **Point** pos)
- void **YachiyoAOI_defineGroup** (int index, double area_xmin, double area_ymin, double area_xmax, double area_ymax, String types)
- void **YachiyoAOI_defineGroup** (int index, **Rectangle** area, String types)
- void **YachiyoAOI_defineMask** (int grpIndex, int maskIndex, double area_xmin, double area_ymin, double area_xmax, double area_ymax, String type)
- void **YachiyoAOI_defineMask** (int grpIndex, int maskIndex, **Rectangle** area, String type)
- void **YachiyoAOI_finish** ()

boolean **YachiyoAOI_generateCalibration** (String name, double pos_x, double pos_y, double startRes, double endRes, double step, String options)
- boolean **YachiyoAOI_generateCalibration** (String name, **Point** pos, double startRes, double endRes, double step, String options)
- boolean **YachiyoAOI_generateOutput** (String name, String lens, String fixture, String options)
- String **YachiyoAOI_getStrings** (String kind)
- boolean **YachiyoAOI_init** (String iniFile)
- void **YachiyoAOI_reset** ()
- void **YachiyoAOI_setRefPoint** (int index, double pos_x, double pos_y)
- void **YachiyoAOI_setRefPoint** (int index, **Point** pos)

Variables

- int **FILE_ATTRIBUTES** = 2
- int **FILE_MODIFICATION_DATE** = 4
- int **FILE_NAME** = 5
- int **FILE_PARENT** = 1
- int **FILE_SIZE** = 3
- int **FILE_TYPE** = 0
- int **LAYER_ACTIVITY** = 5
- int **LAYER_APERTURES** = 6
- int **LAYER_ATTACH** = 3
- int **LAYER_CLASS** = 1
- int **LAYER_INDEX** = 4
- int **LAYER_NAME** = 0
- int **LAYER_SUBCLASS** = 2

Function Documentation

void abort (String *sInfo*)

Aborts script execution

Parameters:

sInfo Text with information about abort reason.

Exceptions:

AbortException

void activate (String *layClass*,
String *laySubclass*,
int *layNum*,
boolean *layAct*
)

Activate/Deactivate layer by class, subclass and index

Parameters:

layClass Class of layer(s) to be activated (layer, drill, extra, feedback or all)

laySubclass Subclass of layer(s) to be activated/deactivated

layNum Index within the specified copper layer class or subclass e.g. to activate the bottom copper layer of a 6 layer job:

```
activate("layer",null , 6, true);
```


```
activate("layer",null , 6, true);
```


The value 6 refers to the 6th layer or

```
activate("layer", "outer", 2, true);
```


```
activate("layer", "outer", 2, true);
```


The value 2 refers to the second layer of subclass "outer". Index within the extra layer class All "extra" layers attached to "top" are numbered from top to bottom, for "extra" layers attached to "bottom" the numbering continues from bottom to top. To activate the extra layer with index 4:

```
activate("extra", null, 4, true);
```


```
activate("extra", null, 4, true);
```


Index within the specified drill class or subclass Drill layers are numbered from left to right To activate the first plated drill layer of a job with 1 unplated and 2 plated layers

```
activate("drill",null , 2, true);
```


```
activate("drill",null , 2, true);
```


The value 2 refers to the second drill layer or

```
activate("drill", "plated", 1, true);
```


```
activate("drill", "plated", 1, true);
```


The value 1 refers to the first plated drill layer **Note:** The `extra.order` ucam.db key can change the order of the extra layers. Scripts relying on the index of extra layers could fail to execute correctly on other systems.

layAct Set true to activate and false to deactivate layers

void activate (ObjectList *layerID,*
boolean *layAct*
)

Activate/Deactivate layer by layer ID

Parameters:

layerID the layer ID describing the layer
layAct Set true to activate and false to deactivate layers

See also:

getLayerID(Ulayer)

getLayerID(Ulayer, String)

void activate (String *layClass,*
String *laySubclass,*
boolean *layAct*
)

Activate/Deactivate layers by subclass

Parameters:

layClass Class of layer(s) to be activated (layer, drill, extra, feedback or all)
laySubclass Subclass of layer(s) to be activated/deactivated
layAct Set true to activate and false to deactivate layers

void activateAllLayers ( )

activateAllLayers

void activateBottomLayers (boolean *layAct*)

Activate/deactivate the bottom layer in all level-1 subjobs.

Parameters:

layAct Set true to activate and false to deactivate layers

void activateToggle ( )

This function can not be recorded Toggle activation of all layers

void activateTopLayers (boolean *layAct*)

Activate/deactivate the top layer in all level-1 subjobs.

Parameters:

layAct Set true to activate and false to deactivate layers

void activityClean (String *sMode*)

Cleans attributes uActivityZoom or uActivityPlane or uActivityActive or all according to given sMode.

Parameters:

sMode string "plane", "zoom", "activity" or "*" for all previous values.

void activityClean ( )

Cleans all attributes uActivityZoom, uActivityPlane and uActivityActive

void activityRestore (String *sMode*)

Restores viewport stored in job attribute uActivityZoom or layer planes of all layers stored in layer attribute named uActivityPlane or layer activity stored in layer attribute named uActivityActive or all according to given sMode.

Parameters:

sMode string "plane", "zoom", "activity" or "*" for all previous values.

void activityRestore ( )

Restores viewport stored in job attribute uActivityZoom, planes of all layers stored in layer attribute named uActivityPlane and layer activity stored in layer attribute named uActivityActive

void activityStore (String *sMode*)

Stores viewport in job attribute uActivityZoom or layer planes of all layers in layer attribute named uActivityPlane or layer activity in layer attribute named uActivityActive or all according to given sMode.

Parameters:

sMode string "plane", "zoom", "activity" or "*" for all previous values.

void activityStore ( )

Stores current viewport in job attribute uActivityZoom, planes of all layers in layer attribute named uActivityPlane and layer activity in layer attribute named uActivityActive

void addBreak (Point *line_fp*,
Point *line_tp*
)

Add break to draws on given line.

Parameters:

line_fp (from point) the given line
line_tp (to point) the given line

void addBreak (double *line_fx*,
double *line_fy*,
double *line_tx*,
double *line_ty*
)

Add break to draws on given line.

Parameters:

line_fx (from X coordinate) the given line
line_fy (from Y coordinate) the given line
line_tx (to X coordinate) the given line
line_ty (to Y coordinate) the given line

void addBreak (Line *line*)

Add break to draws on given line.

Parameters:

line the given line

int addCFMEEAlignmentPoint (double *point_x*,

```
                                    double  point_y
                                )
```

Add alignment point to CFMEE

Parameters:

point_x (X coordinate) the point of the alignment point
point_y (Y coordinate) the point of the alignment point

```
int addCFMEEAlignmentPoint ( Point  point  )
```

Add alignment point to CFMEE

Parameters:

point the point of the alignment point

```
void addDPF ( String  dpf,
              String  layName,
              String  subClass,
              int     layPos,
              String  readable
            )
```

Add DPF Layer

Parameters:

dpf Path to dpf file
layName Name of layer
subClass Subclass of layer
layPos Position of layer
readable Readable side

```
void addDPFDrill ( String  dpf,
                   int     layPos,
                   int     drillTop,
                   int     drillBot
                 )
```

Add DPF Drill

Parameters:

dpf Path to a dpf file
layPos Position of the added layer. The value between 1 and a number of the drill layers inclusive.
drillTop Index of the top drilled layer
drillBot Index of the bottom drilled layer

```
void addDPFExtra ( String  dpf,
                   String  attach,
```

```
                    boolean  bNoChecks
                  )
```

Add DPF Extra

Parameters:

dpf	Path to a dpf file
attach	"top" or "bottom"
bNoChecks	if true, disable overlap check

```
void addDPFExtra ( String  dpf,
                   String  attach
                 )
```

Add DPF Extra

Parameters:

dpf	Path to a dpf file
attach	"top" or "bottom"

```
void addDPFLayer ( String  dpf,
                   int     layPos
                 )
```

Add DPF Layer

Parameters:

dpf	Path to a dpf file
layPos	Position of the added layer. The value between 1 and a number of the signal layers inclusive.

```
int addETMComponentHiPot ( int     etmId,
                           int     primId,
                           int     NetPrim,
                           int     NetSecond,
                           String  TestVolt,
                           String  Duration,
                           String  LeakCurrent,
                           String  VoltType,
                           String  StartVolt,
                           String  VoltRise
                         )
```

Adds a HiPot component to the HiPot etm component and etm connect layers - searches coordinates of largest test point

Parameters:

etmId	The unique etm id of the component
primId	The primary ID for a primary group
NetPrim	Primary net number
NetSecond	Secondary net number

TestVolt	Test parameter Test Voltage [V]
Duration	Test parameter Duration [s]
LeakCurrent	Test parameter Test Leakage Current [mA]
VoltType	Test parameter Voltage Type - AC or DC
StartVolt	Test parameter Start Voltage [V]
VoltRise	Test parameter Voltage Rise [s]

Returns:

0: ok, 1: an error occurred

```
int addETMComponentHiPot ( int    etmId,
                           int    primId,
                           double XStart,
                           double YStart,
                           String AccStart,
                           double XEnd,
                           double YEnd,
                           String AccEnd,
                           int    NetPrim,
                           int    NetSecond,
                           String TestVolt,
                           String Duration,
                           String LeakCurrent,
                           String VoltType,
                           String StartVolt,
                           String VoltRise
                         )
```

Adds a HiPot component to the HiPot etm component and etm connect layers

Parameters:

etmId	The unique etm id of the component
primId	The primary ID for a primary group
XStart	Start coordinate (primary net)
YStart	
AccStart	Connect access for start coordinate (primary net): top or bottom
XEnd	End coordinate (secondary net)
YEnd	
AccEnd	Connect access for end coordinate (secondary net): top or bottom
NetPrim	Primary net number
NetSecond	Secondary net number
TestVolt	Test parameter Test Voltage [V]
Duration	Test parameter Duration [s]
LeakCurrent	Test parameter Test Leakage Current [mA]
VoltType	Test parameter Voltage Type - AC or DC
StartVolt	Test parameter Start Voltage [V]
VoltRise	Test parameter Voltage Rise [s]

Returns:

0: ok, 1: an error occurred

```
void addFault ( String   sType,
                double   oRectangle_xmin,
                double   oRectangle_ymin,
                double   oRectangle_xmax,
                double   oRectangle_ymax,
                String   sInfo
              )
```

Add **Rectangle** as an user fault to current UFD

Parameters:

sType	fault type name
oRectangle_xmin	(left boundary of rectangle) fault **Rectangle**
oRectangle_ymin	(bottom boundary of rectangle) fault **Rectangle**
oRectangle_xmax	(right boundary of rectangle) fault **Rectangle**
oRectangle_ymax	(top boundary of rectangle) fault **Rectangle**
sInfo	fault info

```
void addFault ( String      sType,
                Rectangle   oRectangle,
                String      sInfo
              )
```

Add **Rectangle** as an user fault to current UFD

Parameters:

sType	fault type name
oRectangle	fault **Rectangle**
sInfo	fault info

```
void addFault ( String   sType,
                Line     oLine,
                String   sInfo
              )
```

Add **Line** as an user fault to current UFD

Parameters:

sType	fault type name
oLine	fault **Line**
sInfo	fault info

```
void addFault ( String   sType,
                double   oPt_x,
                double   oPt_y,
                String   sInfo
              )
```

Add point as an user fault to current UFD

Parameters:

sType fault type name
oPt_x (X coordinate) **Point** of the fault
oPt_y (Y coordinate) **Point** of the fault
sInfo fault info

void addFault (String *sType,*
Point *oPt,*
String *sInfo*
)

Add point as an user fault to current UFD

Parameters:

sType fault type name
oPt **Point** of the fault
sInfo fault info

void addHyperScriptMenuItem (String *sScriptPath,*
String *sMenuItemLabel*
)

Adds menu item to Ucam->HyperScript

Parameters:

sScriptPath full path to the script file
sMenuItemLabel menu item label

void addMaskLayer (double *dThicken*)

Create New Soldermask layer

Parameters:

dThicken - Pad Thicken

void addObjectAttribute (String *sAttrName*)

addObjectAttribute Sets the object attribute with the given name. Takes objects from active layers from current job by Ucam options. (e.g. all selected objects)

Parameters:

sAttrName The object attribute name

void addObjectAttribute (String *sAttrName,*
String *sAttrValue*
)

addObjectAttribute Sets the object attribute with the given name and value. Takes objects from active layers from current job by Ucam options. (e.g. all selected objects)

Parameters:

sAttrName The object attribute name
sAttrValue The object attribute value

void addOptimizedMaskLayer (double *dMinRing*,
double *dMaxRing*,
double *dMaskToCopper*,
double *dMaskToMask*,
double *dBigRing*
)

Create New Optimize Soldermask layer

Parameters:

dMinRing - Minimum Ring
dMaxRing - Maximum Ring
dMaskToCopper - Mask to Copper
dMaskToMask - Mask to Mask
dBigRing - Big Pad Ring

void addRefPoint (int *iIndex*,
double *pPnt_x*,
double *pPnt_y*,
boolean *bOnAllActiveLay*
)

Adds a reference point to layer.

Parameters:

iIndex The reference point number.
pPnt_x (X coordinate) The point coordinates.
pPnt_y (Y coordinate) The point coordinates.
bOnAllActiveLay if true it work on all active layers, otherwise only on active loaded layer in plane 1

void addRefPoint (int *iIndex*,
Point *pPnt*,
boolean *bOnAllActiveLay*
)

Adds a reference point to layer.

Parameters:

iIndex The reference point number.
pPnt The point coordinates.
bOnAllActiveLay if true it work on all active layers, otherwise only on active loaded layer in plane 1

```
void addShavedMaskLayer ( double dThicken,
                          double dPadToTrack,
                          double dPadToPad
                        )
```

Create New Soldermask layer and Shave Soldermask

Parameters:

dThicken	- Pad Thicken
dPadToTrack	- Mask to Track
dPadToPad	- Mask To Pad

```
void addTeardrops ( int    iMode,
                    double dRelDiam,
                    double dRelDist,
                    double dAbsDiam,
                    double dAbsDist,
                    double dMinClr,
                    int    iOnRect,
                    int    iOnHoles
                  )
```

Makes teardrop shaped pads where a track enters a cirle/rectangle pad.

Parameters:

iMode	Defines how the teardrop is generated. 0 : using a circular flash. 1 : using draws. 2 : using arcs.
dRelDiam	The relative size of the circular aperture used to create the teardrop. This size is relative to the size of the connected pad in case of sub-lands, or relative to the diameter of the connecting draws/arcs.
dRelDist	The distance of the flash point of the subland relative to the size of the connected pad in sublandmode. This is the length of the bissection relative to the size of the connected pad in track or arc mode.
dAbsDiam	The size of the circular aperture used to create the teardrop.
dAbsDist	The distance of the flash point of the subland in sublandmode, the length on the bissection in track or arc mode.
dMinClr	The minimum clearance.
iOnRect	Generates teardrops on rectangles when 1.
iOnHoles	Generates teardrops only on pads with a drill hole present.

```
void addTeardrops ( int    iMode,
                    double dRelDiam,
                    double dRelDist,
                    double dAbsDiam,
                    double dAbsDist,
                    double dMinClr,
                    int    iOnRect
                  )
```

Makes teardrop shaped pads where a track enters a cirle/rectangle pad.

Parameters:

iMode	Defines how the teardrop is generated. 0 : using a circular flash. 1 : using draws. 2 : using arcs.
dRelDiam	The relative size of the circular aperture used to create the teardrop. This size is relative to the size of the connected pad in case of sub-lands, or relative to the diameter of the connecting draws/arcs.
dRelDist	The distance of the flash point of the subland relative to the size of the connected pad in sublandmode. This is the lengthof the bi-section relative to the size of the connected pad in track or arc mode.
dAbsDiam	The size of the circular aperture used to create the teardrop.
dAbsDist	The distance of the flash point of the subland in sublandmode, the length on the bissection in track or arc mode.
dMinClr	The minimum clearance.
iOnRect	Generates teardrops on rectangles when 1.

int addYsphotechAlignmentPoint (int *region,*
double *point_x,*
double *point_y*
)

Add alignment point to Ysphotech

Parameters:

region	the region number (0 for global)
point_x	(X coordinate) the point of the alignment point
point_y	(Y coordinate) the point of the alignment point

int addYsphotechAlignmentPoint (int *region,*
Point *point*
)

Add alignment point to Ysphotech

Parameters:

region	the region number (0 for global)
point	the point of the alignment point

void align_blocks (double *dTolerance,*
boolean *bOnAllLayers*
)

Performs automatic block align on this job.

Parameters:

dTolerance	required percentage of the overlap
bOnAllLayers	true means ignore layer activity This method is licensed.

void AmliAddUser (String *sUser,*

```
                String  sPassword,
                String  sAuthorityLevel
            )
```

Add user to the system

Parameters:

sUser	user name
sPassword	user's password
sAuthorityLevel	"admin", "operator" or "engineer"

```
void AmliChangeUserPwd ( String  sUser,
                         String  sPassword,
                         String  sNewPassword
                       )
```

Changes user's password.

Parameters:

sUser	user name
sPassword	user password
sNewPassword	user new password

```
String AmliCheckUser ( String  sUser,
                       String  sPassword
                     )
```

Get user's authority level

Parameters:

sUser	user name
sPassword	user password

```
void AmliRemoveUser ( String  sUser )
```

Delete user from the system

Parameters:

sUser	user name to be removed

```
String analyzeExternal ( String  sExtName )
```

Analyze external format of given file

Parameters:

sExtName	external file full path

Returns:

language

void angle (double *angle*)

Set the Angle value

Parameters:

angle Value of the Angle

double angle ( )

Gets the Angle number value

Returns:

Value of the Angle

**void apeAnamorphicScale (double *dDistanceX,*
double *dDistanceY,*
boolean *bProportional*
)**

Anamorphic Scale - Scale X and/or Y size of the aperture of a layer. Only the pad sizes are affected. Anamorphic Scale works on active layer in the plane 1 and on selected objects in the layer.

Parameters:

dDistanceX - a multiplication value, X distance value, absolute
dDistanceY - a multiplication value,Y distance value, absolute
bProportional - if true, the scale will be proportional

**void apeAnamorphicScale (double *dScaleX,*
double *dScaleY*
)**

Anamorphic Scale - Scale X and/or Y size of the aperture of a layer. Only the pad sizes are affected. Anamorphic Scale works on active layer in the plane 1 and on selected objects in the layer.

Parameters:

dScaleX - a multiplication value, X scale value [%]
dScaleY - a multiplication value,Y scale value [%]

**void apeAttribute (String *name,*
String *value*
)**

Sets the given value to the given aperture attribute. If the attribute exists, its value is changed to the new value. Otherwise the attribute is created. If the value is `null` the attribute with the given name is removed.

Parameters:

name the aperture attribute name

value the aperture attribute value

String apeAttribute (String *name*)

Returns the value of the aperture attribute with given name.

Parameters:

name the aperture attribute name

Returns:

the value of the aperture attribute with given name or null if the attribute is not defined in the aperture.

String apeAttribute ( )

Returns comma separated list of the all aperture attributes.

Example: "uPCB=cc_d01,num=1"

Returns:

comma separated list of the all aperture attributes.

void apeCorners (String *sCorners*)

Sets the corner type of the current box aperture.

Parameters:

sCorners String possible values "rounded", "straight", "cut" or "antique".

String apeCorners ( )

Gets the corner type of the current box aperture.

Returns:

"rounded", "straight", "cut" or "antique".

```
void apeCreateBox ( int    apeNum,
                    double xsize,
                    double ysize,
                    String corners,
                    double xcutoff,
                    double ycutoff
                  )
```

Create Box aperture

Parameters:

apeNum Aperture Number
xsize Box size in X

ysize Box size in Y
corners Box corners types : "r" for rounded, "a" for antique, "c" for cut and "s" for straight
xcutoff Box corner cutoff value in X
ycutoff Box corner cutoff value in Y

void apeCreateCircle (int *apeNum,*
double *dia*
)

Create Circle aperture

Parameters:
apeNum Aperture Number
dia Diameter

void apeCreateContour (int *apeNum,*
double *stroke*
)

Create Contour aperture on layer in plane 1

Parameters:
apeNum Aperture Number
stroke Stroke value

void apeCreateDonut (int *apeNum,*
double *outer,*
double *inner,*
String *kind*
)

Create Donut aperture

Parameters:
apeNum Aperture Number
outer Donut outer diameter
inner Donut inner diameter
kind Donut kind : rr=round/round, ss=square/square, sr=square/round

void apeCreateOblong (int *apeNum,*
double *xsize,*
double *ysize*
)

Create Oblong aperture

Parameters:
apeNum Aperture Number

xsize Oblong size in X
ysize Oblong size in Y

void apeCreateOctagon (int *apeNum,*
double *size*
)

Create Octagon aperture

Parameters:
apeNum Aperture Number
size Octagon size

void apeCreateRectangle (int *apeNum,*
double *xsize,*
double *ysize*
)

Create **Rectangle** aperture

Parameters:
apeNum Aperture Number
xsize **Rectangle** size in X
ysize **Rectangle** size in Y

void apeCreateText (int *iApeNum,*
double *dHeight,*
String *sText,*
double *dRotation*
)

Create Text aperture

Parameters:
iApeNum Aperture Number
dHeight Text height
sText Text set to the aperture
dRotation The aperture rotation in degrees. Positive values are counterclockwise.

void apeCreateText (int *iApeNum,*
double *dHeight,*
String *sText*
)

Create Text aperture

Parameters:
iApeNum Aperture Number

dHeight Text height
sText Text set to the aperture

void apeCreateThermal (int *apeNum,*
double *outer,*
double *inner,*
double *gap,*
int *numGap,*
double *angle,*
String *kind*
)

Create Thermal aperture

Parameters:

apeNum Aperture Number
outer Thermal outer diameter
inner Thermal inner diameter
gap Thermal gap distance
numGap Thermal number of gaps
angle Thermal angle
kind Thermal kind : "rr", "rs", "ss" or "sr"

Rectangle apeEnclosingBox ( )

Gets the enclosed rectangle of all objects defined for this aperture.

Returns:

the enclosed rectangle of all objects defined for this aperture.

int apeExtlinkCheck ( )

Check extlink on the current aperture and return a status

Returns:

a status:

```
 0 - OK
 1 - should not occur
 2 - dpf file not found
 3 - cannot load dpf file specified by extlink
 4 - aperture size does not match the dpf file block size
 5 - no external link on the aperture
 6 - no time stamp available in the link
 7 - memory allocation error
 8 - path name cannot be expanded/shrunk
 9 - buffer overrun when expanding/shrinking path name
10 - the file has changed while the definition within the Ucam does not
11 - license check failed
12 - too small buffer to return path name
13 - external link is pointing to itself
14 - time stamp available in the link is different
15 - dpf available in the link is different (dat_equal)
```

String apeExtlinkPath ( )

If the aperture has associated extlink, the pathname is returned

Returns:
external link Path, "" if there is no such link on the aperture

String apeExtlinkPathString ( )

If the aperture has associated extlink, the pathname is returned

Returns:
external link Path, "" if there is no such link on the aperture

boolean apeExtlinkRelative ( )

Returns information if path to external link is relative.

Returns:
true if path to external link is relative.

int apeExtlinkStatus ( )

Take status extlink on the aperture

Returns:
a status:

```
 0 - OK
 1 - should not occur
 2 - dpf file not found
 3 - cannot load dpf file specified by extlink
 4 - aperture size does not match the dpf file block size
 5 - no external link on the aperture
 6 - no time stamp available in the link
 7 - memory allocation error
 8 - path name cannot be expanded/shrunk
 9 - buffer overrun when expanding/shrinking path name
10 - the file has changed while the definition within the Ucam does not
11 - license check failed
12 - too small buffer to return path name
13 - external link is pointing to itself
14 - time stamp available in the link is different
15 - dpf available in the link is different (dat_equal)
```

void apeGap (double *dGap*)

Sets the gap of the current thermal aperture.

Parameters:
dGap double the gap of the current thermal aperture.

double apeGap ( )

Gets the gap of the current thermal aperture.

Returns:
the gap of the current thermal aperture.

boolean apeHasPattern (boolean *bUsed*)

Returns `true` if the aperture has a pattern.

Parameters:

bUsed pattern is used when it affects the image.

Returns:

`true` if the aperture has a pattern.

void apeHeight (double *dHeight*)

Sets the height of the current text aperture.

Parameters:

dHeight double the height of the current text aperture.

double apeHeight ( )

Gets the height of the current text aperture.

Returns:

the height of the current text aperture.

int apeIndex ( )

Gets the index of the current aperture.

Returns:

the index of the current aperture in the layer aperture list.

String apeInfo ( )

Gets the information associated with the current aperture.

Returns:

the information associated with the current aperture. `null` if the current aperture is not set.

void apeInner (double *dInner*)

Sets the inner diameter of the donut or thermal current aperture.

Parameters:

dInner the new inner diameter of the donut or thermal current aperture.

double apeInner ( )

Gets the inner diameter of the donut or thermal current aperture.

Returns:
the inner diameter of the donut or thermal current aperture.

void apeKind (String *sKind*)

Sets the kind of the current thermal aperture.

Parameters:
sKind String "rr" - round-round, "rs" - round-square, "ss" - square-square or "sr" - square-round.

String apeKind ( )

Gets the kind of the current thermal or donut aperture.

Returns:
for Thermal:

- "rr" - round-round
- "rs" - round-square
- "ss" - square-square
- "sr" - square-round. for Donut:
- "RR" : Circular outer and circular inner shape (round/round), the default.
- "SS" : Square outer and square inner shape (square/square).
- "SR" : Square outer and circular inner shape (square/round).

int apeMaxNetNumber ( )

Gets the maximum netnumber in current aperture.

Returns:
the maximum netnumber in current aperture.

void apeMirror (String *sMirror*)

Sets the mirror options for the current aperture.

Parameters:
sMirror String the mirror options for the current aperture. Values "", "X", "Y" or "XY".

String apeMirror ( )

Gets the mirror options for the current aperture.

Returns:
the mirror options for the current aperture. Values "", "X", "Y" or "XY".

void apeName (String *sName*)

Sets the name of the current aperture.

Parameters:

sName String the name of the current aperture.

String apeName ( )

Gets the name of the current aperture.

Returns:

the name of the current aperture.

void apeNumber (int *iNumber*)

Sets the current aperture number.

Parameters:

iNumber int - the new number of the current aperture.

int apeNumber ( )

Gets the current aperture number.

Returns:

the current aperture number.

void apeNumberGap (int *iNumGap*)

Sets the number of gaps in the current thermal aperture.

Parameters:

iNumGap integer - the number of gaps in the current thermal aperture.

int apeNumberGap ( )

Gets the number of gaps in the current thermal aperture.

Returns:

the number of gaps in the current thermal aperture.

int apeNumberObject (String *sObjectClass*)

Gets the number of objects in the current aperture object list. Objects are flashes, draws, arcs and vectortext.

Parameters:

sObjectClass Specifies the class of the objects to count. "f" for flashes, "d" for draws, "a" for arcs and "v" for vectortext.

Returns:
the number of objects in the current aperture object list.

int apeNumberObject ( )

Gets the number of objects in the current aperture object list. Objects are flashes, draws, arcs and vectortext.

Returns:
the number of objects in the current aperture object list.

int apeNumberRegions ( )

Gets the number of regions in the complex, thermal or contour current aperture.

Returns:
the number of regions in the complex, thermal or contour current aperture.

int apeNumContours ( )

Gets the number of contours areas in the complex, thermal or contour current aperture.

Returns:
the number of contours areas in the complex, thermal or contour current aperture.

void apeOuter (double *dOuter*)

Sets the outer diameter of the circle, donut or thermal current aperture.

Parameters:
dOuter double the new diameter of the current aperture.

double apeOuter ( )

Gets the outer diameter of the circle, donut or thermal current aperture.

Returns:
the diameter of the current aperture.

void apePattern (String *sPattern*)

Sets the pattern of the aperture in DPF format.

Parameters:
sPattern String the pattern of the aperture in DPF format.

String apePattern ( )

Gets the pattern of the aperture in DPF format.

Returns:
the pattern of the aperture in DPF format.

void apePatternAngle (double *dPatternAngle*)

Sets the angle of the pattern of the aperture.

Parameters:
dPatternAngle double the angle of the pattern of the aperture in degrees.

double apePatternAngle ( )

Gets the angle of the pattern of the aperture.

Returns:
the angle of the pattern of the aperture.

void apePatternStep (double *dPatternStep*)

Sets the step of the pattern of the aperture.

Parameters:
dPatternStep double - the step of the pattern of the aperture.

double apePatternStep ( )

Gets the step of the pattern of the aperture.

Returns:
the step of the pattern of the aperture.

void apePatternWidth (double *dPatternWidth*)

Sets the width of the pattern of the aperture.

Parameters:
dPatternWidth double the width of the pattern of the aperture.

double apePatternWidth ( )

Gets the width of the pattern of the aperture.

Returns:
the width of the pattern of the aperture.

void apePatternX (double *dX*)

Sets the x size of the pattern of the aperture.

Parameters:
dX double the x size of the pattern of the aperture in degrees.

double apePatternX ( )

Gets the x size of the pattern of the aperture.

Returns:
the x size of the pattern of the aperture.

void apePatternY (double *dY*)

Sets the y size of the pattern of the aperture.

Parameters:
dY double the y size of the pattern of the aperture in degrees.

double apePatternY ( )

Gets the y size of the pattern of the aperture.

Returns:
the y size of the pattern of the aperture.

Rectangle apeRectangle ( )

Gets the size of the aperture definition as a rectangle.

Returns:
the size of the aperture definition as a rectangle.

void apeReverse (boolean *bReverse*)

Sets the reverse status for the current aperture.

Parameters:
bReverse boolean `true` for reverse, `false` otherwise.

boolean apeReverse ( )

Gets the reverse status for the current aperture.

Returns:
`true` for reverse, `false` otherwise.

void apeRotation (double *dRotation*)

Sets the rotation of the current aperture.

Parameters:
dRotation double - the rotation of the current aperture.

double apeRotation ( )

Gets the rotation of the current aperture.

Returns:
the rotation of the current aperture.

void apeScale (double *dScale*)

Sets the scale factor of the current aperture.

Parameters:
dScale double - the scale factor of the current aperture.

double apeScale ( )

Gets the scale factor of the current aperture.

Returns:
the scale factor of the current aperture.

boolean apeSelection ( )

Checks if the current aperture contains selected items.

Returns:
1 if there are selections, 0 otherwise.

Rectangle apeSelectionEnclosingBox ( )

Gets the enclosed rectangle of all selected objects in current aperture.

Returns:
the enclosed rectangle of all selected objects in current aperture.

String apeShape ( )

Gets the shape of the current aperture.

Returns:
the shape of the current aperture ("cir","blo","box","com","con","don","rec","txt","the","oct","und").

void apeSize (double *dSize*)

Sets the size of the current octagon aperture.

Parameters:
dSize double the size of the current octagon aperture.

double apeSize ( )

Gets the size of the current octagon aperture.

Returns:
the size of the current octagon aperture.

void apeStartAngle (double *dStartAngle*)

Sets the start angle of the current thermal aperture.

Parameters:
dStartAngle double - the start angle of the current thermal aperture.

double apeStartAngle ( )

Gets the start angle of the current thermal aperture.

Returns:
the start angle of the current thermal aperture.

void apeString (String *sString*)

Sets the text of the current text aperture.

Parameters:
sString String - the text of the current text aperture.

String apeString ( )

Gets the text of the current text aperture.

Returns:
the text of the current text aperture.

void apeStroke (double *dStroke*)

Sets the stroke width for the current contour aperture.

Parameters:
dStroke double - the stroke width for the current contour aperture.

double apeStroke ( )

Gets the stroke width for the current contour aperture.

Returns:
the stroke width for the current contour aperture.

double apeSurface ( )

Get the surface of the current contour aperture in squared current units.

Returns:
the surface of the current contour aperture in squared current units.

void apeThickenThin (double *value,*
boolean *keepArcs*
)

Perform thicken or thin (spread or choke) function on apertures. Aperture size will be changed.

Parameters:
value The value to thicken (positive) or thin (negative)
keepArcs If true, arcs are maintained within complex apertures.

void apeWidth (double *dWidth*)

Sets the width of the current text aperture.

Parameters:
dWidth double - the width of the current text aperture.

double apeWidth ( )

Gets the width of the current text aperture.

Returns:

the width of the current text aperture.

void apeXCutOff (double *dXCutOff*)

Sets the x cutoff value for the corners of the current box aperture.

Parameters:

dXCutOff double - the x cutoff value for the corners.

double apeXCutOff ( )

Gets the x cutoff value for the corners of the current box aperture.

Returns:

the x cutoff value for the corners.

void apeXSize (double *dXSize*)

Sets the x size of the rectangle or box current aperture.

Parameters:

dXSize double - the x size of the current aperture.

double apeXSize ( )

Gets the x size of the rectangle or box current aperture.

Returns:

the x size of the current aperture.

void apeYCutOff (double *dYCutOff*)

Sets the y cutoff value for the corners of the current box aperture.

Parameters:

dYCutOff double - the y cutoff value for the corners.

double apeYCutOff ( )

Gets the y cutoff value for the corners of the current box aperture.

Returns:

the y cutoff value for the corners.

void apeYSize (double *dYSize*)

Sets the y size of the rectangle or box current aperture.

Parameters:

dYSize double - the y size of the current aperture.

double apeYSize ( )

Gets the y size of the rectangle or box current aperture.

Returns:

the y size of the current aperture.

int applyHorns (String *hornType,*
double *minimumClearance,*
ObjectList *params*
)

Applies horns to all hornable pads in current selection or in current layer if there is no selection.

Parameters:

hornType	the type of horn to apply. Can be "up", "up_tilted", "side", "corner_rec", "corner_square" or "corner_flat". (See drawings at [we need a location to add drawings, this is impossible to explain otherays]).
minimumClearance	minimum free distance to other objects after applying horns
params	an array containing all the parameters needed for the specific horn type. (See drawings at [we need a location to add drawings, this is impossible to explain otherays]).

Returns:

amount of pads to which horns where applied if successful -1 if params array contains non-floats -2 if an invalid horn type has been given -3 if params array was not of the right length

Arc Arc (double *ptFromX,*
double *ptFromY,*
double *ptToX,*
double *ptToY,*
double *ptCenterX,*
double *ptCenterY,*
String *sSense,*
String *sUnits*
)

Create arc from six coordinates and sense and units

Parameters:

ptFromX	start point x coordinate
ptFromY	start point y coordinate
ptToX	end point x coordinate
ptToY	end point y coordinate
ptCenterX	center point x coordinate
ptCenterY	center point y coordinate

sSense arc sense
sUnits Ucam units

Returns:
the arc

Arc Arc (double *ptFromX*,
double *ptFromY*,
double *ptToX*,
double *ptToY*,
double *ptCenterX*,
double *ptCenterY*,
String *sSense*
)

Create arc from six coordinates and sense

Parameters:
ptFromX start point x coordinate
ptFromY start point y coordinate
ptToX end point x coordinate
ptToY end point y coordinate
ptCenterX center point x coordinate
ptCenterY center point y coordinate
sSense arc sense

Returns:
the arc

Arc Arc (Arc *oArc*)

Create copy of a arc

Parameters:
oArc original arc

Returns:
the arc

Arc Arc (Point *ptFrom*,
Point *ptTo*,
Point *ptCenter*,
String *sSense*
)

Create arc from three points and sense

Parameters:
ptFrom arc start point
ptTo arc end point
ptCenter arc end point

sSense arc sense

Returns:
the arc

void autofixtureBuildFixture (boolean *bFixtureBuild,*
String *sFixture*
)

Sets parameters to dialog Autofixture

Parameters:
bFixtureBuild enable Fixture setting
sFixture Fixture setting

Exceptions:
AbortException

void autofixtureDo ( )

Generate all data for an Autofixture

void autofixtureMicroAdjustment (boolean *bMicroAdjustment,*
int *iNbrOfTestPoints,*
double *dTestPointDiameter,*
double *dTestPointShiftEdge,*
double *dTestPointShiftValue,*
double *dTestPointPitch,*
double *dClearanceFactor,*
double *dCenterDiameter*
)

Sets parameters to dialog Autofixture and to subdialog Micro Adjustment

Parameters:

bMicroAdjustment	enable Micro Adjustment Setup
iNbrOfTestPoints	The following alignment point parameters are used for all selected test points.
dTestPointDiameter	The aperture diameter of the pads used as alignment points (in the current unit).
dTestPointShiftEdge	The distance between the test point and the first alignment point's center (in the current unit).
dTestPointShiftValue	The distance between each of the alignment points' center on the axis of the shortest side (in the current unit).
dTestPointPitch	The distance between each of the alignment points center on the axis of the longest side (in the current unit).
dClearanceFactor	The minimum clearance required between other copper areas and the edge of the test point (at the longest side).
dCenterDiameter	The aperture diameter of the center point (in the current unit).

void autofixtureNetlist (boolean *bNetlist,*

```
                    boolean  bNetlistBuild,
                    boolean  bNetlistExpand
                )
```

Sets parameters to dialog Autofixture to section Netlist

Parameters:

bNetlist	if sets true, generate the netlist for the current job.
bNetlistBuild	if sets true, generates or regenerates a netlist of the job.
bNetlistExpand	if sets true, generates the netlist for a panelized job

```
void autofixtureOutput ( boolean  bOutput  )
```

Sets parameters to dialog Autofixture

Parameters:

bOutput	enable output settings

```
void autofixtureTestpoints ( boolean  bTestPoints,
                             int      iLoop,
                             boolean  bUseMasks,
                             boolean  bProbeSwaping,
                             boolean  bHandlePaintedPads,
                             boolean  bCircuitryCheck,
                             boolean  bFilterCopperAreas,
                             boolean  bViaOfSMDs,
                             boolean  bDrillsWithoutPad
                           )
```

Sets parameters to dialog Autofixture to section Testpoints

Parameters:

bTestPoints	if true, calculate the test points of a job and to create one or two test point layers for the job.
iLoop	sets how to test pads in a loop
bUseMasks	if true, takes all solder mask layers into account for test point calculation.
bProbeSwaping	if true, marks all test points that can be technically tested on the other side of the pcb.
bHandlePaintedPads	if thue, handle painted pads
bCircuitryCheck	if true,enables the generation of test points according to electrical test Optimization Rules.
bFilterCopperAreas	if true, reduces the number of test points generated in large coppers by removing all unnecessary test points that are satisfactorily surrounded by copper.
bViaOfSMDs	if true, generates the test points only on the via holes of SMD's, according to the attribute settings for uVia.
bDrillsWithoutPad	if true, generates test points on drill holes without pad.

```
void autofixtureTestpointsBot ( boolean  bPointsBot1,
                                boolean  bPointsBot2,
```

```
                                boolean  bPointsBot3,
                                boolean  bPointsBot4,
                                boolean  bPointsBot5,
                                boolean  bPointsBot6,
                                boolean  bPointsBot7
                              )
```

Sets parameters to dialog Autofixture to section Testpoints - Optimization Rule bottom side

Parameters:

bPointsBot1 Sets a test point on a pad that is a drilled pad and that is not connected with any track (on the top or bottom layer).

bPointsBot2 Sets a test point on a pad that is a drilled pad and that is connected with only one track on the test point side and not connected with any track on the opposite layer.

bPointsBot3 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with only one track on the opposite layer.

bPointsBot4 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to one inner layer.

bPointsBot5 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with more than one track on the opposite layer.

bPointsBot6 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to two or more inner layers.

bPointsBot7 Sets a test point on a pad that is not a drilled pad and that is connected with more than one track.

```
void autofixtureTestpointsTop ( boolean  bPointsTop1,
                                boolean  bPointsTop2,
                                boolean  bPointsTop3,
                                boolean  bPointsTop4,
                                boolean  bPointsTop5,
                                boolean  bPointsTop6,
                                boolean  bPointsTop7
                              )
```

Sets parameters to dialog Autofixture to section Testpoints - Optimization Rule top side

Parameters:

bPointsTop1 Sets a test point on a pad that is a drilled pad and that is not connected with any track (on the top or bottom layer).

bPointsTop2 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with only one track on the opposite layer.

bPointsTop3 Sets a test point on a pad that is a drilled pad and that is connected with only one track on the test point side and not connected with any track on the opposite layer.

bPointsTop4 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to one inner layer.

bPointsTop5 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with more than one track on the opposite layer.

bPointsTop6 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to two or more inner layers.

bPointsTop7 Sets a test point on a pad that is not a drilled pad and that is connected with more than one track.

void blockEdit ( )

Aperture Manager: Enters Block Definition Edit Mode for current Block Aperture

void blockMultiEdit ( )

Aperture Manager: Enters Block Definition Multi Edit Mode for current Block Aperture

int BlockReconstruct ( )

Makes a block out of the current selection of the layer in plane1 and replaces any occurrences of the same data with a block flash.

Returns:

a negative number if a problem was detected or the number of blocks that were created otherwise.

**void boardSnapshot (boolean *graph,*
String *templPath,*
boolean *pio,*
String *pioPath*
)**

Generate Board Snapshot

Parameters:

graph	Generate graphical output if true
templPath	Path to template files
pio	Generate Product information text output if true
pioPath	Path to information text

void buildSubJobs ( )

Created dafault sub-jobs

**void calculateImpedance (String *sImpConfig,*
ObjectList *parameters*
)**

Calculate impedance

Parameters:

sImpConfig	impedance configuration
parameters	parameter array, e.g. {[width, height, thickness, er]}

boolean canRead (ObjectList *fileInfo*)

Tests whether the application can read the file denoted by this fileInfo.

Parameters:

fileInfo objectlist with the file information

Returns:

`true` if and only if the file denoted by this file info and it is allowed to read from the file; `false` otherwise

See also:

HSH_base::osFileInfo(String)

boolean canWrite (ObjectList *fileInfo*)

Tests whether the application can modify the file denoted by this fileInfo.

Parameters:

fileInfo objectlist with the file information

Returns:

`true` if and only if the file denoted by this file info and it is allowed to write to the file; `false` otherwise

See also:

HSH_base::osFileInfo(String)

void center (double *center_x*,
double *center_y*
)

Sets the Center Used in Numbers Dialog

Parameters:

center_x (X coordinate) the Center

center_y (Y coordinate) the Center

void center (Point *center*)

Sets the Center Used in Numbers Dialog

Parameters:

center the Center

Point center ( )

Gets the Center Used in Numbers Dialog

Returns:

Center

void centerX (double *centerX*)

Sets the Center x coordinate Used in Numbers Dialog

Parameters:

centerX the Center x coordinate

void centerY (double *centerY*)

Sets the Center y coordinate Used in Numbers Dialog

Parameters:

centerY the Center y coordinate

void chain ( )

If the order of the draws in a chain is not consecutive or if some of the draws have their vector-direction reversed you can use Chain to make the (selected) chains continuous.

void chamferJoin (double *pt_x*,
double *pt_y*,
double *disX*,
double *disY*
)

Chamfer (cut the corner of) existing draws.

Parameters:

pt_x (X coordinate) The junction of the two draws
pt_y (Y coordinate) The junction of the two draws
disX Horizontal length value (from chamfer to join)
disY Vertical length value (from join to chamfer)

void chamferJoin (Point *pt*,
double *disX*,
double *disY*
)

Chamfer (cut the corner of) existing draws.

Parameters:

pt The junction of the two draws
disX Horizontal length value (from chamfer to join)
disY Vertical length value (from join to chamfer)

void changeDirection (double *p_x*,
double *p_y*
)

Changes direction of the closest rout chain. The point is defined as X and Y world coordinates.

Parameters:

p_x (X coordinate) point where to look for the rout chain
p_y (Y coordinate) point where to look for the rout chain

void changeDirection (Point *p*)

Changes direction of the closest rout chain. The point is defined as X and Y world coordinates.

Parameters:

p point where to look for the rout chain

int changePrioPlotQueue (String *sRipHost,*
int *iJobId,*
int *iPriority*
)

Change plot queue job priority

Parameters:

sRipHost name of the Rip host
iJobId ID of the rip job
iPriority new priority

Returns:

0 if OK, otherwise is an error

boolean checkDrillInfo (boolean *bSelNonPlated,*
boolean *bAssignAttributes,*
boolean *bBlocksOnly*
)

Drill Info Generates drill info and selects non plated holes in the active layers.

Parameters:

bSelNonPlated When true, non plated drill holes are selected.
bAssignAttributes When true, the drill info results are stored as attributes UdrillStat on the drill holes
bBlocksOnly When true, drill info is only calculated for objects in Panel StepRepeat blocks

Returns:

a status, false = ok

String chooseDirPath (String *sTitle,*
String *sStartDir*
)

Opens Open File dialog and let the user select directory path

Parameters:

sTitle Dialog title
sStartDir Starting directory

Returns:
Directory path

Exceptions:
AbortException After Cancel button the script is aborted.

String chooseDirPath (String *sTitle*)

Opens Open File dialog and let the user select directory path

Parameters:
sTitle Dialog title

Returns:
Directory path

Exceptions:
AbortException After Cancel button the script is aborted.

String chooseDirPath ( )

Opens Open File dialog and let the user select directory path

Returns:
Directory path

Exceptions:
AbortException After Cancel button the script is aborted.

String chooseFilePath (String *sTitle,*
String *sStartDir,*
String *sFileMask*
)

Opens Open File dialog and let the user select file path **Example:**

```
openJob(chooseFilePath("Select Job", "D:/MyJobs/Test", "*.job"));
openJob(chooseFilePath("Select Job", "D:/MyJobs/Test", "*.job"));
```

Parameters:
sTitle Dialog title
sStartDir initial directory when Open dialog is opened
sFileMask file name mask

Returns:
Selected file path

Exceptions:
AbortException After Cancel button the script is aborted.

See also:

chooseFilePath()

chooseFilePath()

chooseFilePath(String)

chooseFilePath(String)

chooseFilePath(String, String)

chooseFilePath(String, String)

String chooseFilePath (String *sTitle*)

Opens Open File dialog and let the user select file path

Parameters:

sTitle Dialog title

Returns:

Selected file path

Exceptions:

AbortException After Cancel button the script is aborted.

See also:

chooseFilePath(String, String, String)

chooseFilePath(String, String, String)

**String chooseFilePath (String *sStartDir,*
String *sFileMask*
)**

Opens Open File dialog and let the user select file path

Parameters:

sStartDir initial directory when Open dialog is opened

sFileMask file name mask

Returns:

Selected file path

Exceptions:

AbortException After Cancel button the script is aborted.

See also:

chooseFilePath(String, String, String)

chooseFilePath(String, String, String)

String chooseFilePath ( )

Opens Open File dialog and let the user select file path

Returns:

Selected file path

Exceptions:

AbortException After Cancel button the script is aborted.

See also:

chooseFilePath(String, String, String)

chooseFilePath(String, String, String)

void cleanApertures ( )

Aperture Manager: Group Apertures by Polarity in active layer in plane 1

void cleanApeTables ( )

Clean aperture tables on all active layers

void cleanETMComponentLayers (int *type*)

Creates or cleans existing component layers

Parameters:

type The type of the component layers to clean: 0: CAPACITOR; 1: INDUCTOR; 2: KELVIN; 3: RESISTOR; 4: DIODE; 5: HIPOT

void cleanSubJobs ( )

Clean all sub-jobs

void cleanUfd (String *sUfdName*)

Create new empty fault database with the given name and add it to Error Manager

Parameters:

sUfdName fault database name

void cleanUnderBlo ( )

Clean data under BLO with uPcb attribute using CLIPPING and MERGE.

void cleanup (double *dReconstructArcs,*
double *dValidateArcs,*
double *dRemoveObsoleteObjects,*

```
                double  dRemoveSmallObjects,
                double  dReconnectObjects,
                boolean bReconstructArcs,
                boolean bValidateArcs,
                boolean bRemoveObsoleteObjects,
                boolean bRemoveSmallObjects,
                boolean bReconnectObjects
              )
```

Parameters:

dReconstructArcs	arc reconstruct tolerance
dValidateArcs	arc validation tolerance
dRemoveObsoleteObjects	remove obsolete objects tolerance
dRemoveSmallObjects	remove small objects tolerance
dReconnectObjects	reconnect objects tolerance
bReconstructArcs	when true, arcs are reconstructed
bValidateArcs	when true, arcs are validated
bRemoveObsoleteObjects	when true, obsolete objects are removed
bRemoveSmallObjects	when true, small objects are removed
bReconnectObjects	when true, objects are reconnected

void clearance (double *clearance*)

Set the Clearance value

Parameters:

clearance Value of the Clearance

double clearance ( )

Gets the Clearance number value

Returns:

Value of the Clearance

```
void clearanceCheckMAT ( double  dPadSpread,
                         double  dSmdSpread,
                         double  dTrackSpread,
                         double  dAreaSpread,
                         double  dPadPadClearance,
                         double  dPadSmdClearance,
                         double  dPadTrackClearance,
                         double  dPadAreaClearance,
                         double  dSmdSmdClearance,
                         double  dSmdTrackClearance,
                         double  dSmdAreaClearance,
                         double  dTrackTrackClearance,
                         double  dTrackAreaClearance,
```

```
                    double  dAreaAreaClearance,
                    boolean bCheckSameNetSpacing,
                    boolean bFastMode,
                    int     iShiftMode,
                    double  dMinCopper
                )
```

Secure Etch Compensation

Parameters:

dPadSpread	The spread value to apply to circular pads
dSmdSpread	The spread value to apply to non circular pads
dTrackSpread	The spread value to apply to draws
dAreaSpread	The spread value to apply to areas
dPadPadClearance	The clearance to keep between circular pads
dPadSmdClearance	The clearance to keep between circular pads and non circular pads
dPadTrackClearance	The clearance to keep between circular pads and draws
dPadAreaClearance	The clearance to keep between circular pads and areas
dSmdSmdClearance	The clearance to keep between non circular pads
dSmdTrackClearance	The clearance to keep between non circular pads and draws
dSmdAreaClearance	The clearance to keep between non circular pads and areas
dTrackTrackClearance	The clearance to keep between draws
dTrackAreaClearance	The clearance to keep between draws and areas
dAreaAreaClearance	The clearance to keep between areas
bCheckSameNetSpacing	Whether to check spacing between objects of the same net as well
bFastMode	If true, skip slow "select embedded" step to make check faster but sometimes less correct
iShiftMode	Mode to shift clearances, 0: no shift, 1: shift to cut only areas, 2: like 1, but keep original copper of areas
dMinCopper	The minimum copper width to keep

```
void clearanceCheckMAT ( double  dPadSpread,
                         double  dSmdSpread,
                         double  dTrackSpread,
                         double  dAreaSpread,
                         double  dPadPadClearance,
                         double  dPadSmdClearance,
                         double  dPadTrackClearance,
                         double  dPadAreaClearance,
                         double  dSmdSmdClearance,
                         double  dSmdTrackClearance,
                         double  dSmdAreaClearance,
                         double  dTrackTrackClearance,
                         double  dTrackAreaClearance,
                         double  dAreaAreaClearance,
                         boolean bCheckSameNetSpacing,
                         boolean bFastMode,
                         int     iShiftMode
                )
```

Secure Etch Compensation

Parameters:

dPadSpread	The spread value to apply to circular pads
dSmdSpread	The spread value to apply to non circular pads
dTrackSpread	The spread value to apply to draws
dAreaSpread	The spread value to apply to areas
dPadPadClearance	The clearance to keep between circular pads
dPadSmdClearance	The clearance to keep between circular pads and non circular pads
dPadTrackClearance	The clearance to keep between circular pads and draws
dPadAreaClearance	The clearance to keep between circular pads and areas
dSmdSmdClearance	The clearance to keep between non circular pads
dSmdTrackClearance	The clearance to keep between non circular pads and draws
dSmdAreaClearance	The clearance to keep between non circular pads and areas
dTrackTrackClearance	The clearance to keep between draws
dTrackAreaClearance	The clearance to keep between draws and areas
dAreaAreaClearance	The clearance to keep between areas
bCheckSameNetSpacing	Whether to check spacing between objects of the same net as well
bFastMode	If true, skip slow "select embedded" step to make check faster but sometimes less correct
iShiftMode	Mode to shift clearances, 0: no shift, 1: shift to cut only areas, 2: like 1, but keep original copper of areas

```
void clearanceCheckMAT ( double  dPadSpread,
                         double  dSmdSpread,
                         double  dTrackSpread,
                         double  dAreaSpread,
                         double  dPadPadClearance,
                         double  dPadSmdClearance,
                         double  dPadTrackClearance,
                         double  dPadAreaClearance,
                         double  dSmdSmdClearance,
                         double  dSmdTrackClearance,
                         double  dSmdAreaClearance,
                         double  dTrackTrackClearance,
                         double  dTrackAreaClearance,
                         double  dAreaAreaClearance,
                         boolean bCheckSameNetSpacing,
                         boolean bFastMode
                       )
```

Secure Etch Compensation

Parameters:

dPadSpread	The spread value to apply to circular pads
dSmdSpread	The spread value to apply to non circular pads
dTrackSpread	The spread value to apply to draws
dAreaSpread	The spread value to apply to areas
dPadPadClearance	The clearance to keep between circular pads
dPadSmdClearance	The clearance to keep between circular pads and non circular pads
dPadTrackClearance	The clearance to keep between circular pads and draws
dPadAreaClearance	The clearance to keep between circular pads and areas
dSmdSmdClearance	The clearance to keep between non circular pads
dSmdTrackClearance	The clearance to keep between non circular pads and draws

dSmdAreaClearance	The clearance to keep between non circular pads and areas
dTrackTrackClearance	The clearance to keep between draws
dTrackAreaClearance	The clearance to keep between draws and areas
dAreaAreaClearance	The clearance to keep between areas
bCheckSameNetSpacing	Whether to check spacing between objects of the same net as well
bFastMode	If true, skip slow "select embedded" step to make check faster but sometimes less correct

void clearMessages ( )

Clears messages window

boolean clipping (int *iClipReference,*
String *sClipSide,*
double *dClipClr,*
double *dMinLineLength,*
boolean *bRounded*
)

Clipping

Parameters:

iClipReference	Ref layer for clipping (0 for outline, 1, 2, 3 or 4 for plane colors)
iClipReference	Ref layer for clipping (0 for outline, 1, 2, 3 or 4 for plane colors)
sClipSide	Side where clipping should be applied ("outside" or "inside")
dClipClr	Clearance after clipping
dMinLineLength	Minimum length of clipped objects
bRounded	True if endpoints of clipped objects should be round

Returns:

status value

boolean clipSilk (double *dClr,*
double *dMinLen*
)

Silk Optimize Clips all silk layers in this job. The corresponding mask layers is used as a reference for clipping, or the corresponding outer layer is used in case there is no mask layer found. Only pads and tracks are clipped in the silk layers, and only pads are used as clipping reference.

Parameters:

dClr	is the clearance value that should be used for clipping. Silk Clipping removes data from the silk layer so that silk clearance is assured between objects of the silk layer and objects of the reference layer.
dMinLen	indicates the value for which tracks smaller than this value will be removed.

Returns:

true if no clipping was done, if no problem then returns false.

void closeAMLIJobManager ( )

close AMLI Job Manager

void closeAnamorphicScale ( )

close AnamorphicScale dialog

void closeApeCreator ( )

close Aperture Creator

void closeApeEditor ( )

close Aperture Editor

void closeApertureAttributes ( )

close Aperture Attributes dialog

void closeApertureManager ( )

close Aperture Manager dialog

void closeAttributeEditor ( )

close Attribute Editor dialog

void closeAttributeManager ( )

close Attribute Manager dialog

void closeAutoDrill ( )

close AutoDrill dialog

void closeAutoDrillEditor ( )

Close AutoDrill Editor

void closeAutoFixture ( )

close AutoFixture dialog

void closeBarcode ( )

close Barcode dialog

void closeBarcode128 ( )

close Barcode 128 dialog

void closeBoardAnalyzer ( )

close Board Analyzer dialog

void closeBoardSnapshot ( )

close Board Snapshot dialog

void closeCalculatorSetup ( )

close Calculator Setup dialog

void closeCamtek ( )

close Camtek dialog

void closeCheckList ( )

close CheckList Dialog

void closeCheckListDefineChecklist ( )

Close "CheckList: Define Checklist" Dialog

void closeCheckListDefineSteps ( )

Close "CheckList: Define Steps" Dialog

void closeClipping ( )

close Clipping dialog

void closeColor ( )

Close Color dialog

void closeConnect ( )

close Connect dialog

void closeContourHandling ( )

close Contour Handling dialog

void closeConvertAttributes ( )

close Attribute Converter dialog

void closeCopperBalance ( )

close Copper Balance Dialog

void closeCopperRepair ( )

close Copper Repair dialog

void closeCoverlayOptimizer ( )

close Coverlay Optimizer dialog

void closeCU9000Dialog ( )

Close DS DI output dialog

void closeDatums ( )

close Datums dialog

void closeDistort ( )

close Distort dialog

void closeDrawSlots ( )

close Draw Slots Dialog

void closeDRC ( )

close DRC dialog

void closeDrillInfo ( )

close Drill Info dialog

void closeDrillMap ( )

close Drill Map Dialog

void closeDrillOptimizer ( )

close Smart Drill Optimizer

void closeDrillRoutSetups ( )

Close Drill/Rout Setups dialog

void closeDrillTolerance ( )

Close Drill Tolerance dialog

void closeDrillToolManager ( )

close Drill Tool Manager

void closeDsAoi ( )

Close DS AOI dialog

void closeDSAOIialog ( )

Close DS DI output dialog

void closeDsAoiPreview ( )

Close DS AOI dialog

void closeEditingToolbox ( )

close Editing Toolbox dialog

void closeEditVectorText ( )

close Edit Vector Text dialog

void closeErrors ( )

close Errors dialog

void closeEtchCompensation ( )

close Etch Compensation dialog

void closeExpand ( )

close Expand dialog

void closeExternalLinkManager ( )

close External Link Manager

void closeFiducials ( )

close Fiducials dialog

void closeFillAngledPattern ( )

close Fill Angled Pattern Dialog

void closeFillPattern ( )

close Fill Pattern Dialog

void closeFillVector ( )

close Fill Vector Dialog

void closeFlashMaker ( )

close FlashMaker dialog

void closeFlexManager ( )

close uFlex Manager

void closeFlipJob ( )

close Flip Job dialog

void closeFrame (String *sFrameName*)

Closes custom dockable frame with given identification

Parameters:

sFrameName identification frame given by getFrameID() method of CustomFrame class

void closeGridParameters ( )

close Grid Parameters dialog

void closeHiPot ( )

close HiPot dialog

void closeImageCompare ( )

close Image Compare dialog

void closeImpedanceControl ( )

close Impedance Control dialog

void closeImportODBxx ( )

close Import ODBxx Steps dialog

void closeInsertContourText ( )

close Insert Contour Text dialog

void closeInsertVectorText ( )

close Insert Vector Text dialog

void closeJobDefinition ( )

close Job Definition dialog

void closeJobEdit ( )

close Job Edit dialog

void closeJobEditor ( )

Close Job Editor dialog

void closeJobEditorOptions ( )

Close Job Editor Options dialog

void closeJobMerge ( )

close Job Merge dialog

void closeJobPlaneSetup ( )

Close Job Plane Setup dialog

void closeJobPrint ( )

Close Job Print dialog

void closeLayerEdit ( )

close Layer Modify dialog

void closeLegendOptimizer ( )

close Legend Optimizer dialog

void closeLoadCheckList ( )

close Load CheckList Dialog

void closeMagnifier ( )

close Magnifier window

void closeMarkupAssistant ( )

close Markup Assistant

void closeMessages ( )

close Messages log window

void closeMLIOutput ( )

close MLI Output dialog

void closeModels ( )

close Models dialog

void closeNetCompare ( )

close Net Compare dialog

void closeNonFunctionalPad ( )

close Non-Functional pads dialog

void closeNumbers ( )

close Numbers dialog

void closeObjectAttributes ( )

close Object Attributes dialog

void closeObjectCompare ( )

close Object Compare dialog

void closeOutputAccumatch ( )

Close Accumatch Output dialog

void closeOutputAOI ( )

Close AOI Output dialog

void closeOutputCAD ( )

close Output CAD dialog

void closeOutputCamtek ( )

Close Camtek Output dialog

void closeOutputDrillRout ( )

close Drill/Rout dialog

void closeOutputDsDi ( )

Close DS DI output dialog

void closeOutputDsDiPreview ( )

Close DS DI Preview dialog

void closeOutputNetlist ( )

close Output Netlist dialog

void closeOutputOrbot ( )

Close Orbot Output dialog

void closeOutputSapphire ( )

Close Sapphire Output dialog

void closeOutputScoring ( )

close Output Scoring dialog

void closeOutputSmartArgos ( )

Close SmartArgos Output dialog

void closeOutputTrackscan ( )

Close Trackscan Output dialog

void closeOutputUxpAutomanager ( )

Close Output UXP Automanager dialog

void closeOutputUxpEtec ( )

Close Output UXP Etec dialog

void closePanelFramesCoupons ( )

Close Panel Frames Coupons dialog

void closePanelLinks ( )

Close Panel Links dialog

void closePanelPlus ( )

close PanelPlus dialog

void closePanelReproduce ( )

close PanelReproduce dialog

void closePanelSetup ( )

Close Panel Setup dialog

void closePanelStepRepeat ( )

close Panel Step Repeat dialog

void closePlaneAdjuster ( )

close Plane Adjuster dialog

void closePlotParameters ( )

close Plot Parameters dialog

void closePPMonitor ( )

close PPMonitor dialog

void closeQueryNet ( )

close Query Net dialog

void closeQueryObject ( )

close Query Object dialog

void closeReferencePoints ( )

close Reference Points dialog

void closeRegister ( )

close Register dialog

void closeRemoveAttributes ( )

close Remove Attributes dialog

void closeRepair ( )

Close Repair dialog

void closeRoutManager ( )

close Rout Editor dialog

void closeRoutManagerCleanUp ( )

close Rout Editor dialog

void closeRoutManagerDimensioning ( )

close Rout Editor dialog

void closeRoutManagerEditor ( )

close Rout Editor dialog

void closeRoutManagerTools ( )

close Rout Editor dialog

void closeSaveLayout ( )

close Save Window Layout dialog

void closeSecureEtchCompensation ( )

close Secure Etch Compensation dialog

void closeSelections ( )

close Selections dialog

void closeSetupOptions ( )

Close Setup Options dialog

void closeSetupSave ( )

Close Save dialog

void closeShavePads ( )

close Shave Pads dialog

void closeSignalLayerAdjuster ( )

close Signal Layer Adjuster dialog

void closeSignalLayerAdjusterAssistant ( )

close Signal Layer Adjuster Assistant dialog

void closeSilkOptimizer ( )

close Silk Optimizer

void closeSmartCamtek ( )

close SmartCamtek dialog

void closeSmartDRC ( )

close Smart DRC dialog

void closeSmartFix ( )

close SmartFix dialog

void closeSmartplot ( )

Close Smartplot dialog

void closeSmartSR ( )

Close Smart S&R dialog

void closeSmartStart ( )

close Smart Start dialog

void closeSoldermask ( )

close Soldermask Dialog

void closeSoldermaskOptimizer ( )

close Soldermask Optimizer Dialog

void closeTearDrop ( )

close Tear Drop Dialog

void closeTechnicalAnalyzer ( )

close Technical Analyzer dialog

void closeTestpointEdit ( )

close Testpoint edit dialog

void closeToolbarManager ( )

close Toolbar Manager Dialog

void closeToolbars ( )

close Toolbars Dialog

void closeTransformObjects ( )

close Transform Objects dialog

void closeTransformObjectsBGAPads ( )

close Transform Objects BGA Pads dialog

void closeTransformObjectsBGATracks ( )

close Transform Objects BGA Tracks dialog

void closeTransformObjectsEdit ( )

close Transform Objects Edit dialog

void closeTransformObjectsRescale ( )

close Transform Objects Rescale dialog

void closeUcamDbEditor ( )

Close Ucamdb Editor dialog

void closeUndoRedoDetails ( )

close Undo/Redo Details

void closeUTest ( )

close UTest dialog

void closeUtestUtilities ( )

close Utest Utilities dialog

void closeValidateLayer ( )

close Layer Validation dialog (or not, if everything is fine)

void closeVectorTextFont ( )

close Vector Text Font dialog

void closeVerifyArcsDraws ( )

close Verify Arcs and Draws dialog

void closeViewGuide ( )

close View Guide dialog

void colorAll (String *exclSubClass,
boolean *bKeepLayActivity*
)

Assigns a plane to all layers that have no plane assigned yet.

Parameters:

exclSubClass null or comma separated string, list subclasses that won't be colored.

bKeepLayActivity `true` keeps layer activity, `false` deactivates all layers without color if active.

void colorAll (String *exclSubClass*)

Assigns a plane to all layers that have no plane assigned yet.

Parameters:

exclSubClass null or comma separated string, list subclasses that won't be colored.

void compareImage (String *reference*,
boolean *bAutoAlign*,
double *missingTol*,
double *exceedingTol*,
int *iErrorAccuracy*,
ObjectList *revPolArr*,
int *compSelMode*
)

Image Compare (external reference)

Parameters:

reference	Reference job or reference layer to compare with
bAutoAlign	Automatically align current and reference layers
missingTol	Allowed tolerance of missing copper
exceedingTol	Allowed tolerance of exceeding copper
iErrorAccuracy	The display accuracy for differences
revPolArr	Array of layers with reverse polarity, e.g. [{["lay1", false, true}], [{"lay2", true, true}]}] means: lay1 of current job not reversed but reference reversed; lay2 current and reference both reversed
compSelMode	0: compare all, 1: blocks - compare on primary, 2: compare in areas, 3: compare outside areas

void compareImage (String *reference*,
boolean *bAutoAlign*,
double *missingTol*,
double *exceedingTol*,
int *iErrorAccuracy*,
ObjectList *revPolArr*
)

Image Compare (external reference)

Parameters:

reference	Reference job or reference layer to compare with
bAutoAlign	Automatically align current and reference layers
missingTol	Allowed tolerance of missing copper
exceedingTol	Allowed tolerance of exceeding copper
iErrorAccuracy	The display accuracy for differences
revPolArr	Array of layers with reverse polarity, e.g. [{["lay1", false, true}], [{"lay2", true, true}]}] means: lay1 of current job not reversed but reference reversed; lay2 current and

reference both reversed

```
void compareImage ( double  missingTol,
                    double  exceedingTol,
                    int     iErrorAccuracy,
                    boolean bRevPolarityCur,
                    boolean bRevPolarityRef,
                    int     compSelMode
                  )
```

Image Compare (Layer 1 - Layer 2)

Parameters:

missingTol	Allowed tolerance of missing copper
exceedingTol	Allowed tolerance of exceeding copper
iErrorAccuracy	The display accuracy for differences
bRevPolarityCur	If true, layer 1 (current) has reversed polarity,
bRevPolarityRef	If true, layer 2 (reference) has reversed polarity,
compSelMode	0: compare all, 1: blocks - compare on primary, 2: compare in areas, 3: compare outside areas

```
void compareImage ( double  missingTol,
                    double  exceedingTol,
                    int     iErrorAccuracy,
                    boolean bRevPolarityCur,
                    boolean bRevPolarityRef
                  )
```

Image Compare (Layer 1 - Layer 2)

Parameters:

missingTol	Allowed tolerance of missing copper
exceedingTol	Allowed tolerance of exceeding copper
iErrorAccuracy	The display accuracy for differences
bRevPolarityCur	If true, layer 1 (current) has reversed polarity,
bRevPolarityRef	If true, layer 2 (reference) has reversed polarity,

```
void compareNet ( int     iMode,
                  boolean bCheckFlash
                )
```

Net Compare

Parameters:

iMode	Compare mode - 1: basic tests, 2: 1 + multiple nets, 3: 2 + not fully covered
bCheckFlash	true: test for missing flash

```
void compareNet ( int     iMode,
                  boolean bCheckFlash,
                  String  sReferenceFile,
                  boolean bPanelize
                )
```

Net Compare

Parameters:

iMode	Compare mode - 1: basic tests, 2: 1 + multiple nets, 3: 2 + not fully covered
bCheckFlash	true: test for missing flash
sReferenceFile	Reference file for multi job net compare
bPanelize	Panelize reference

```
void compareNet ( int     iMode,
                  boolean bCheckFlash,
                  boolean bIgnoreOutline,
                  double  dOutlineMargin,
                  boolean bIgnoreNPTHPads,
                  double  dNPTHExpandMargin,
                  String  sReferenceFile,
                  boolean bPanelize
                )
```

Net Compare

Parameters:

iMode	Compare mode - 1: basic tests, 2: 1 + multiple nets, 3: 2 + not fully covered
bCheckFlash	true: test for missing flash
bIgnoreOutline	true: ignore nets outside the outline
dOutlineMargin	Net references inside the outline and near the outline edge within the chosen margin are also ignored.
bIgnoreNPTHPads	true: ignore for the 'Lost Elements' test those pads which correspond to a NPTH hole
dNPTHExpandMargin	Net references which correspond to a NTPH hole after expanding the hole with this margin are ignored
sReferenceFile	Reference file for multi job net compare
bPanelize	Panelize reference

Deprecated:

use **compareNet(boolean, boolean, boolean, boolean, boolean, boolean, boolean, double, boolean, double, String, boolean, boolean)**

use **compareNet(boolean, boolean, boolean, boolean, boolean, boolean, boolean, double, boolean, double, String, boolean, boolean)**

```
void compareNet ( boolean bShorts,
                  boolean bOpens,
                  boolean bLostElements,
                  boolean bDoubleNet,
                  boolean bNotCovered,
                  boolean bCheckFlash,
```

```
                boolean bIgnoreOutline,
                double  dOutlineMargin,
                boolean bIgnoreNPTHPads,
                double  dNPTHExpandMargin,
                String  sReferenceFile,
                boolean bPanelize,
                boolean bBuildNetlist
              )
```

Net Compare

Parameters:

bShorts	true: test for shorts
bOpens	true: test for opens
bLostElements	true: test for missing copper elements
bDoubleNet	true: test for reference pads touching more than one net
bNotCovered	true: test for reference pads not fully covered by copper
bCheckFlash	true: test for missing flash
bIgnoreOutline	true: ignore nets outside the outline
dOutlineMargin	Net references inside the outline and near the outline edge within the chosen margin are also ignored.
bIgnoreNPTHPads	true: ignore for the 'Lost Elements' test those pads which correspond to a NPTH hole
dNPTHExpandMargin	Net references which correspond to a NTPH hole after expanding the hole with this margin are ignored
sReferenceFile	Reference file for reference job comparison
bPanelize	Panelize reference
bBuildNetlist	Build Job Netlist

```
void compareObjects ( String  referenceJob,
                      double  xTol,
                      double  yTol,
                      boolean bWindow,
                      boolean bObjMoved,
                      boolean bObjAdded,
                      boolean bObjNet,
                      boolean bApeShape,
                      boolean bApeSize,
                      boolean bApeOrder
                    )
```

Object Compare

Parameters:

referenceJob	Reference job to campare with
xTol	Tolerance of comparison in X direction
yTol	Tolerance of comparison in Y direction
bWindow	Reference points in drill layer are used to define area
bObjMoved	Mark objects that have been moved
bObjAdded	Mark objects that exist only in current job
bObjNet	Check net numbers
bApeShape	Check for changes to aperture shapes

bApeSize Check for changes to aperture sizes
bApeOrder Check order in aperture list

void compensate (String *sSense,*
double *dis*
)

Compensate rout for the thickness of the routing tool.

Parameters:
sSense Compensates for the size of the routing tool to the "left" or "right"
dis Distance of compensation

void complexEdit ( )

Aperture Manager: Enters Complex Definition Edit Mode for current Complex Aperture

int connectPadTrack (double *dActiveRadius,*
double *dSnapRadius,*
boolean *bUseNetlist*
)

connectPadTrack

Parameters:
dActiveRadius Maximum length of connect vector
dSnapRadius Maximum distance over which a vector can be snapped
bUseNetlist If true only vectors of identical, valid net can be connected

Returns:
error number (100 = no netlist) or number of errors

void connectTracks ( )

Connect tracks corresponds to: Transform Objects - BGA Tracks - Connect Tracks

boolean contourizeBitmap (int *iPpi,*
double *dMargin,*
double *dDxdy,*
double *dDx,*
double *dDy*
)

contourizeBitmap

Parameters:
iPpi

dMargin
dDxdy
dDx
dDy

Returns:
status

void contourizeExact ( )

Exact Contourize analytic contourize

void contourizeExactAperture ( )

Exact Contourize the (current) aperture

boolean contourizePatterns (int *iPpi*)

Contourize apertures with patterns in the layer in plane 1

Parameters:
iPpi given DPI

Returns:
status false = 0 = ok;

boolean contourizePatternsinJob (int *iPpi*)

Contourize apertures with patterns in all active layers

Parameters:
iPpi given DPI

Returns:
status false = 0 = ok;

void contourThickenThin (double *value,*
boolean *bKeepArcs*
)

Perform thicken or thin (spread or choke) function on contours. Aperture size will be changed.

Parameters:
value The value to thicken (positive) or thin (negative)
bKeepArcs If true, arcs are maintained.

boolean convertGar (String *sInputFile,*

```
                String  sGarFile,
                String  sOutputFile
            )
```

convert input file by rules from gar file to output file

Parameters:

sInputFile	full path to input file
sGarFile	full path to gar file (rules)
sOutputFile	full path to output file

Returns:

status

```
void copperBalancePad ( double  dMinClrToCopper,
                        double  dMinClrToBoard,
                        double  dMinConSurface,
                        String  sFillPattern,
                        double  dPatternClr,
                        double  dApeSize,
                        String  sApeShape
                      )
```

Creates the venting pattern. The area will be filled with pads

Parameters:

dMinClrToCopper	- Minimum Clearance to Copper, specify the clearance between the copper from the affected layer and the newly created pattern. If the layer is drilled, this value will also be used as clearance between the copper pattern and the holes.
dMinClrToBoard	- Minimum Clearance to Board. Specify the clearance from the board edge to the outline of the pattern contour. The original outline will be shrinked with this value to meet the desired clearance.
dMinConSurface	- Minimum Contour Surface. All the copper surfaces, smaller than the specified value, are removed.
sFillPattern	- Choose one of the following options: - full - pads are placed on each grid points - even - pads are placed only on even grid points - odd - pads are placed only on odd grid points
dPatternClr	- Specify the grid step used to place the pads or tracks.
dApeSize	- Specify the size of the aperture used to fill the area.
sApeShape	- Choose one of the following options - circle, hexagon, diamond

```
void copperBalanceSolid ( double  dMinClrToCopper,
                          double  dMinClrToBoard,
                          double  dMinConSurface,
                          double  dApeSize
                        )
```

Creates the venting pattern. The area will be filled with a solid contour

Parameters:

dMinClrToCopper	- Minimum Clearance to Copper, specify the clearance between the copper from the affected layer and the newly created pattern. If the layer is drilled, this value will also be used as clearance between the copper pattern and the holes.

dMinClrToBoard - Minimum Clearance to Board. Specify the clearance from the board edge to the outline of the pattern contour. The original outline will be shrinked with this value to meet the desired clearance.

dMinConSurface - Minimum Contour Surface. All the copper surfaces, smaller than the specified value, are removed.

dApeSize - Specify the size of the aperture used to fill the area.

```
void copperBalanceTrack ( double  dMinClrToCopper,
                          double  dMinClrToBoard,
                          double  dMinConSurface,
                          String  sLineStyle,
                          double  dPatternClr,
                          double  dApeSize,
                          double  dRotation
                        )
```

Creates the venting pattern. The area will be filled with tracks using a circle aperture

Parameters:

dMinClrToCopper - Minimum Clearance to Copper, specify the clearance between the copper from the affected layer and the newly created pattern. If the layer is drilled, this value will also be used as clearance between the copper pattern and the holes.

dMinClrToBoard - Minimum Clearance to Board. Specify the clearance from the board edge to the outline of the pattern contour. The original outline will be shrinked with this value to meet the desired clearance.

dMinConSurface - Minimum Contour Surface. All the copper surfaces, smaller than the specified value, are removed.

sLineStyle - Select the track style to be applied: "parallel lines" or "crosshatched by lines"

dPatternClr - Specify the grid step used to place the pads or tracks.

dApeSize - Specify the size of the aperture used to fill the area.

dRotation - set a rotation angle for tracks

```
void copperCount ( String  sOpt )
```

Deprecated:

Copper count without mask layer usage

Parameters:

sOpt Set to "job", "layer", or "inner"

```
void copperRepair ( String  sOpt,
                    double  smallerThan,
                    double  minSize,
                    double  expand
                  )
```

Copper Repair

Parameters:

sOpt Repair option ("pinholes", "peelables" or "slivers")

smallerThan	pinholes, peelables or slivers smaller than this value are repaired
minSize	peelables or slivers larger than this value are repaired
expand	Expand value for pinholes, peelables or slivers

void copy (double *pt_x*,
double *pt_y*
)

Duplicate (selected) object(s) using board coordinates

Example:

```
setInPlane(1,1);
direction("");
copy(100,200);
copy(100,200);
doCopy(100,200);

direction("h");
copy(100,200);
copy(100,200);
doCopy(100,200);

direction("v");
copy(100,200);
copy(100,200);
doCopy(100,200);
```

Parameters:

pt_x (X coordinate) Offset (vector) where to create the copy of the source objects

See also:

com.barco.ets.ucam.hypershell.HyperShell::doCopy(Upoint)

com.barco.ets.ucam.hypershell.HyperShell::direction(String)

Parameters:

pt_y (Y coordinate) Offset (vector) where to create the copy of the source objects

See also:

com.barco.ets.ucam.hypershell.HyperShell::doCopy(Upoint)

com.barco.ets.ucam.hypershell.HyperShell::direction(String)

void copy (Point *pt*)

Duplicate (selected) object(s) using board coordinates

Example:

```
setInPlane(1,1);
direction("");
copy(100,200);
copy(100,200);
doCopy(100,200);

direction("h");
copy(100,200);
copy(100,200);
doCopy(100,200);

direction("v");
copy(100,200);
copy(100,200);
doCopy(100,200);
```

Parameters:

pt Offset (vector) where to create the copy of the source objects

See also:

com.barco.ets.ucam.hypershell.HyperShell::doCopy(Upoint)

com.barco.ets.ucam.hypershell.HyperShell::direction(String)

void copyOutline (double *refPoint_x*,
double *refPoint_y*,
double *offset_x*,
double *offset_y*,
double *rotation*
)

PCB images in a flat data need to be outlined. We can manually construct outline contour in this (usually the layer in plane 1 and outline extra layer) layer according to a PCB image in reference layer. We mark reference point in PCB image and give an offset of the same reference data in second PCB image we want to outline. The rotation says the PCB outline on target PCB should be rotated 0,90,180,270 degree.

Parameters:

refPoint_x	(X coordinate) A point where the PCB data are taken as an reference (for alignment)
refPoint_y	(Y coordinate) A point where the PCB data are taken as an reference (for alignment)
offset_x	(X coordinate) a target PCB should have the same objects (from reference point) at the offset position
offset_y	(Y coordinate) a target PCB should have the same objects (from reference point) at the offset position
rotation	the target Outline has to be rotated with given angle (0,90,180,270)

Returns:

Uapeobj can be `null` in case the new outline couldn't be created. The return aperture can be the same in case the target is without rotation. Only the object(flash) is the new. In case there is a rotation the return object is completely new aperture with the (one) new flash.

void copyOutline (Point *refPoint*,
Point *offset*,
double *rotation*
)

PCB images in a flat data need to be outlined. We can manually construct outline contour in this (usually the layer in plane 1 and outline extra layer) layer according to a PCB image in reference layer. We mark reference point in PCB image and give an offset of the same reference data in second PCB image we want to outline. The rotation says the PCB outline on target PCB should be rotated 0,90,180,270 degree.

Parameters:

refPoint	A point where the PCB data are taken as an reference (for alignment)
offset	a target PCB should have the same objects (from reference point) at the offset position
rotation	the target Outline has to be rotated with given angle (0,90,180,270)

Returns:

Uapeobj can be `null` in case the new outline couldn't be created. The return aperture can be the same in case the target is without rotation. Only the object(flash) is the new. In case there is a rotation the return object is completely new aperture with the (one) new flash.

void copyToClipboard ( )

Copy all (selected) objects on layer in plane 1 Objects are stored in clipboard

void coreInfo ( )

Core/Prepreg Info Gives information to log window (console) Adds information like attribute to dpf and job files

int countAmbiguousContours ( )

Count ambiguous contours.

Returns:
count of ambiguous contours

int countAmbiguousContoursOnLayer ( )

Count ambiguous contours.

Returns:
count of ambiguous contours

int countInvalidArcs ( )

Count invalid arcs.

Returns:
count of ambiguous contours

int countInvalidArcsOnLayer ( )

Count invalid arcs.

Returns:
count of ambiguous contours

int countInvalidDraws ( )

Count invalid draws.

Returns:
count of invalid draws

int countInvalidDrawsOnLayer ( )

Count Invalid Draws on layer.

Returns:
count of invalid draws

int countOpenContours ( )

Count Open Contours.

Returns:
count of open contours

int countOpenContoursOnLayer ( )

Count Open Contours on layer.

Returns:
count of open contours

int countOverlapContours ( )

Count Overlap Contours.

Returns:
count of overlap contours

int countOverlapContoursOnLayer ( )

Count Overlap Contours on layer.

Returns:
count of overlap contours

int countUndefinedApertures ( )

Count Undefined apertures.

Returns:
count of undefined apertures

int countUndefinedAperturesOnLayer ( )

Count Undefined apertures on lauer.

Returns:
count of undefined apertures

```
int countZeroDrawsArcs ( double   dMaxLength,
                         boolean  bFunctional,
                         boolean  bNonFunctional
                       )
```

Count ZeroDrawsArcs.

Parameters:

dMaxLength	Maximum length of objects to be selected
bFunctional	Select objects within functional copper if true
bNonFunctional	Select objects within non-functional copper if true

Returns:

count of ZeroDrawsArcs

```
int countZeroDrawsArcsOnLayer ( double   dMaxLength,
                                boolean  bFunctional,
                                boolean  bNonFunctional
                              )
```

Count ZeroDrawsArcs on layer.

Parameters:

dMaxLength	Maximum length of objects to be selected
bFunctional	Select objects within functional copper if true
bNonFunctional	Select objects within non-functional copper if true

Returns:

count of ZeroDrawsArcs

```
void countZeroLengthDrawsArcs ( double   dMaxLength,
                                boolean  bFunctional,
                                boolean  bNonFunctional
                              )
```

Count Zero Length Draws and Arcs

Parameters:

dMaxLength	Maximum length of objects to be selected
bFunctional	Select objects within functional copper if true
bNonFunctional	Select objects within non-functional copper if true

```
void createAperture ( int         iApeNum,
                      String      sApeName,
                      String      sApeDef,
                      ObjectList  attrArray
                    )
```

Aperture Manager: Create an Aperture

Parameters:

iApeNum	Number of Aperture to create; if < 0, next free number will be used
sApeName	Name of Aperture to create
sApeDef	DPF style Definition String of Aperture, e.g. "REC,1.905,0.3048"
attrArray	Array of Aperture Attributes, e.g. [{"attr1=v1", "attr2=", "attr3=v3"}]

boolean createBarcode128 (double *dHeight,*
double *dNarrowX,*
String *sValue*
)

Create barcode 128 like block aperture and add the aperture to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

dHeight	The barcode height.
dNarrowX	The width of narrow bars.
sValue	a string, what will be transform tu barcode

Returns:

status, if return true, the function was ok

boolean createBarcode39 (double *dHeight,*
double *dNarrowX,*
double *dRatio,*
String *sValue*
)

Create barcode 39 like block aperture and add the aperture like current to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

dHeight	The barcode height.
dNarrowX	The width of narrow bars.
dRatio	The barcode ratio.
sValue	a string, what will be transform tu barcode

Returns:

status, if return true, the function was ok

boolean createBarcodeInterleaved25 (double *dHeight,*
double *dNarrowX,*
String *sValue*
)

Create barcode interleaved 25 like block aperture and add the aperture like current to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

dHeight	The barcode height.
dNarrowX	The width of narrow bars.

sValue a string, what will be transform tu barcode

Returns:

status, if return true, the function was ok

```
void createBlockAperture ( int        iApeNum,
                           String     sApeName,
                           String     sApeDef,
                           ObjectList attrArray,
                           int        iMode,
                           boolean    bWithCenter,
                           double     pt_x,
                           double     pt_y,
                           String     sExtFile
                         )
```

Aperture Manager: Create a Block Aperture

Parameters:

iApeNum	Number of Aperture to create (if < 0, next free number will be used)
sApeName	Name of Aperture to create
sApeDef	DPF style Definition String of Aperture; 'BLO' and 'size' is not required
attrArray	Array of Aperture Attributes, e.g. [{"attr1=v1", "attr2=", "attr3=v3"}]
iMode	0: empty block, 1: use selections, 2: link to external dpf file
bWithCenter	subtract center offset (used only for iMode 1)
pt_x	(X coordinate) center
pt_y	(Y coordinate) center
sExtFile	External link file (used only for iMode 2)

```
void createBlockAperture ( int        iApeNum,
                           String     sApeName,
                           String     sApeDef,
                           ObjectList attrArray,
                           int        iMode,
                           boolean    bWithCenter,
                           Point      pt,
                           String     sExtFile
                         )
```

Aperture Manager: Create a Block Aperture

Parameters:

iApeNum	Number of Aperture to create (if < 0, next free number will be used)
sApeName	Name of Aperture to create
sApeDef	DPF style Definition String of Aperture; 'BLO' and 'size' is not required
attrArray	Array of Aperture Attributes, e.g. [{"attr1=v1", "attr2=", "attr3=v3"}]
iMode	0: empty block, 1: use selections, 2: link to external dpf file
bWithCenter	subtract center offset (used only for iMode 1)
pt	center
sExtFile	External link file (used only for iMode 2)

```
void createComplexAperture ( int        iApeNum,
                             String     sApeName,
                             String     sApeDef,
                             ObjectList attrArray,
                             boolean    bUseRegion,
                             boolean    bWithCenter,
                             double     pt_x,
                             double     pt_y
                           )
```

Aperture Manager: Create a Complex Aperture

Parameters:

iApeNum	Number of Aperture to create (if < 0, next free number will be used)
sApeName	Name of Aperture to create
sApeDef	DPF style Definition String of Aperture; 'COM' and 'size' is not required
attrArray	Array of Aperture Attributes, e.g. [{"attr1=v1", "attr2=", "attr3=v3"}]
bUseRegion	true: use only regions, false: use all selections
bWithCenter	Subtract center offset
pt_x	(X coordinate) center point
pt_y	(Y coordinate) center point

```
void createComplexAperture ( int        iApeNum,
                             String     sApeName,
                             String     sApeDef,
                             ObjectList attrArray,
                             boolean    bUseRegion,
                             boolean    bWithCenter,
                             Point      pt
                           )
```

Aperture Manager: Create a Complex Aperture

Parameters:

iApeNum	Number of Aperture to create (if < 0, next free number will be used)
sApeName	Name of Aperture to create
sApeDef	DPF style Definition String of Aperture; 'COM' and 'size' is not required
attrArray	Array of Aperture Attributes, e.g. [{"attr1=v1", "attr2=", "attr3=v3"}]
bUseRegion	true: use only regions, false: use all selections
bWithCenter	Subtract center offset
pt	center point

```
void createDataMatrix ( String sTextToEncode,
                        String sMode,
                        String sFormat,
                        double dDotSize,
                        double dRotation,
                        String sMirror,
```

```
                    double  dClearance,
                    boolean bReverse
                )
```

Create barcode - data matrix like block aperture and add the aperture like current to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

sTextToEncode	Text of the DataMatrix
sMode	Mode of the encoding (MODE_ASCII, MODE_C40, MODE_TEXT, MODE_BASE256, MODE_NONE, MODE_AUTO)
sFormat	Format of the DataMatrix
dDotSize	Size of the dot in the DataMatrix
dRotation	rotation
sMirror	The mirror setting, either "", "X", "Y" or "XY".
dClearance	clearance
bReverse	Indication whether the block aperture needs to be reversed or not.

```
void createDrill ( String layName,
                   String subClass,
                   int    from,
                   int    to
                 )
```

Create Drill

Parameters:

layName	Name of layer
subClass	Subclass of layer
from	From layer
to	To layer

```
void createDrill ( int laynum,
                   int drillFrom,
                   int drillTo
                 )
```

create drill layer

Parameters:

laynum	The number of the drill layer
drillFrom	Index of the top drilled layer
drillTo	Index of the bottom drilled layer

```
void createExtra ( String layName,
                   String subClass,
                   String attach,
                   int    index
                 )
```

Create Extra Layer

Parameters:

layName	Name of layer
subClass	Subclass of layer
attach	Attached to top, bottom or none
index	Layer index

**void createExtra (String *layName,*
String *subClass,*
String *attach*
)**

Create Extra Layer

Parameters:

layName	Name of layer
subClass	Subclass of layer
attach	Attached to top, bottom or none

void createExtra (String *attach*)

Create Extra Layer

Parameters:

attach	"top" or "bottom"

**void createLayer (String *layName,*
String *subClass,*
int *layPos,*
String *readable*
)**

Create Layer

Parameters:

layName	Name of layer
subClass	Subclass of layer
layPos	Position of layer
readable	Readable side

void createLayer (int *laynum*)

create signal layer

Parameters:

laynum	The number of the signal layer

```
void createQRCode ( String  sCode,
                    double  dDotSize,
                    double  dAngle,
                    String  sMirror,
                    double  dClr,
                    double  dLabelClr,
                    String  sLabelPos,
                    boolean bMicroQR,
                    boolean bReverse
                  )
```

Create QRCode as a block aperture and add the aperture like current to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

sCode	Text of the QRCode
dDotSize	Size of the dot in the QRCode
dAngle	Rotation
sMirror	The mirror setting, either "", "X", "Y" or "XY".
dClr	Clearance
dLabelClr	Label clearance
sLabelPos	label position possible values are "top" or "bottom"
bMicroQR	if true the QRCode is MicroQR
bReverse	Indication whether the block aperture needs to be reversed or not.

```
void createQRCode ( String  sCode,
                    double  dDotSize,
                    double  dAngle,
                    String  sMirror,
                    double  dClr,
                    String  sLabel,
                    double  dLabelClr,
                    String  sLabelPos,
                    boolean bMicroQR,
                    boolean bReverse
                  )
```

Create QRCode as a block aperture and add the aperture like current to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

sCode	Text of the QRCode
dDotSize	Size of the dot in the QRCode
dAngle	Rotation
sMirror	The mirror setting, either "", "X", "Y" or "XY".
dClr	Clearance
sLabel	The QRCode label.
dLabelClr	Label clearance
sLabelPos	label position possible values are "top" or "bottom"
bMicroQR	if true the QRCode is MicroQR
bReverse	Indication whether the block aperture needs to be reversed or not.

void createQRCode (String *sCode,*
double *dDotSize,*
double *dAngle,*
String *sMirror,*
double *dClr,*
boolean *bMicroQR,*
boolean *bReverse*
)

Create QRCode as a block aperture and add the aperture like current to the layer. If you want add a flash, you must call insertFlash too.

Parameters:

sCode	Text of the QRCode
dDotSize	Size of the dot in the QRCode
dAngle	Rotation
sMirror	The mirror setting, either "", "X", "Y" or "XY".
dClr	Clearance
bMicroQR	if true the QRCode is MicroQR
bReverse	Indication whether the block aperture needs to be reversed or not.

void createSubJob (int *from,*
int *to,*
int *selectedSubJob,*
int *selectedLevel*
)

Modify/Create sub-job

Parameters:

from	layer index the new or modified sub-job starts at
to	layer index the new or modified sub-job ends at
selectedSubJob	the index of the sub-job, starts with 0, -1 - create new sub-job
selectedLevel	the index of the level of the sub-job, starts with 0, -1 - create new sub-job

void createVoronoiDiagram (int *iEdgeTypes*)

Create a new layer which contains the voronoi diagram of the layer in plane 1. This method works on selections, too. This method is licensed.

Parameters:

iEdgeTypes	bit mask specifying which edge/node types need to be output: INNER_EDGES = 0x01; OUTER_EDGES = 0x02; MAIN_EDGES = 0x04 (net separators); DEGENERATE_EDGES = 0x08; BIG_DEADENDS = 0x10; MAXIMUM_NODES = 0x20; ALL_EDGES = 0x3f.

void createVoronoiDiagram (int *iEdgeTypes,*
boolean *bExpandArcs*

)

Create a new layer which contains the voronoi diagram of the layer in plane 1. This method works on selections, too. This method is licensed.

Parameters:

iEdgeTypes	bit mask specifying which edge/node types need to be output: INNER_EDGES = 0x01; OUTER_EDGES = 0x02; MAIN_EDGES = 0x04 (net separators); DEGENERATE_EDGES = 0x08; BIG_DEADENDS = 0x10; MAXIMUM_NODES = 0x20; ALL_EDGES = 0x3f.
bExpandArcs	Expand arcs before calculating the diagram.

**void createVoronoiEdgesExtFile (boolean *bExpandArcs,*
String *sFilePath,*
String *sOptions*
)**

Create an external file with the voronoi diagram edges of the layer in plane 1. This method works on selections, too. This method is licensed.

Parameters:

bExpandArcs	Expand arcs before calculating the diagram.
sFilePath	complete destination file path
sOptions	options separated by comma like: binary(default), xml, copper_only, space_only (default is copper and space)

void CU9000ApplyPlotstamps (ObjectList *plotstamps*)

The function takes the Object Aray with plotststmps' definitions and applies the changes to plotstamps linked to given layer (usually layer in plane 1)

Parameters:

plotstamps	Object Array with plotstams' definitions

boolean CU9000CheckPlotstamps ( )

The method checks Level plotstamps if they are place at correct Level (PCB counter at PCB level etc.)

Returns:

`false` check failed due to invalid conditions (layer is not in plane 1, license is missing etc.)

**boolean CU9000DetectAutoAreas (String *resultLayerName,*
String *referenceLayerName,*
double *margin*
)**

Automatically generate rectangular areas from defined area marks in reference layer.

Parameters:

resultLayerName	name of the generated area layer

referenceLayerName	reference layer name where area marks are presented
margin	margin to be added

Returns:

false if a problem was detected during Area Detection or true otherwise

**boolean CU9000DetectExactAreas (String *resultLayerName,*
int *blockMode,*
String *pcbName,*
String *referenceLayerName,*
double *margin,*
double *outline*
)**

Automatically generate exact outline areas for uPCB blocks for the red layer

Parameters:

resultLayerName	name of the generated area layer
blockMode	what to use as a reference for block detection: OUTLINE (1) - use outline layer REFERENCE_LAYER (2) - use specified reference layer SELF (3) - use red layer
pcbName	uPCB name to search (if empty, search for deepest level)
referenceLayerName	reference layer name REFERENCE_LAYER block mode
margin	margin to be added
outline	stroke width for exact outline detection

Returns:

false if a problem was detected during Area Detection or true otherwise.

int CU9000DetectGlobalAlignment (String *sAPRefLayerName*)

Automatically detects global alignment points in the red layer

Parameters:

sAPRefLayerName	a layer name containing alignment points

Returns:

a negative number if a problem was detected or the number of placed global alignment points otherwise

int CU9000DetectGlobalAlignment ( )

Automatically detects global alignment points in the red layer

Returns:

a negative number if a problem was detected or the number of placed global alignment points otherwise

int CU9000DetectLocalAlignmentPoints (String *sAPRefLayerName*)

Automatically detects local alignment points in the red layer

Parameters:

sAPRefLayerName a layer name containing alignment points

Returns:

a negative number if a problem was detected or the number of placed local alignment points otherwise

int CU9000DetectLocalAlignmentPoints ( )

Automatically detects local alignment points in the red layer

Returns:

a negative number if a problem was detected or the number of placed local alignment points otherwise

boolean CU9000DetectRectangularAreas (String *resultLayerName,* int *blockMode,* String *pcbName,* String *referenceLayerName,* double *margin*)

Automatically generate rectangular areas for uPCB blocks for the red layer

Parameters:

resultLayerName	name of the generated area layer
blockMode	what to use as a reference for block detection: OUTLINE (1) - use outline layer REFERENCE_LAYER (2) - use specified reference layer SELF (3) - use red layer
pcbName	uPCB name to search (if empty, search for deepest level)
referenceLayerName	reference layer name REFERENCE_LAYER block mode
margin	margin to be added

Returns:

false if a problem was detected during Area Detection or true otherwise

ObjectList CU9000GetPlotstamps ( )

The function returns Object Array of the plotstamps' definitions for given (current) layer.

Returns:

Object Array with plotstamps' definitions related to the given layer. **Example:** [{[{"Code=PP,Polarity=Positive,Mirror=x,Rotation=0.0", "247.3172", "170.7196", "panel=1,array=3,pcb=3"}], [{"Code=PP,Polarity=Positive,Mirror=x,Rotation=0.0", "247.3172", "205.7196", "panel=1,array=3,pcb=1"}], [{"Code=PP,Polarity=Positive,Mirror=x,Rotation=0.0", "282.3172", "170.7196", "panel=1,array=3,pcb=4"}], ...}]}]

void CU9000GUIApply ( )

Method simulates Apply button press.

void CU9000GUILoadAlignment (String *sAlignmentPath*)

loads Alignment points definition file and updates GUI

Parameters:

sAlignmentPath Alignment points definition file (TXT or Gerber) full path

void CU9000GUILoadBrd (String *sBrdPath*)

loads definition file and updates GUI

Parameters:

sBrdPath Board Definition file (.brd) full path

void CU9000GUILoadRgi (String *sRgiPath*)

loads definition file and updates GUI

Parameters:

sRgiPath Board Definition file (.rgi) full path

void CU9000GUISaveAlignment (String *sAlignmentPath*)

saves Alignment points definition file

Parameters:

sAlignmentPath Alignment points definition file (TXT or Gerber) full path

boolean CU9000LoadBoardSetup (String *path*)

Apply the given Board Setup file to all active layers

Parameters:

path path to board setup file

Returns:

false if the board file could not be loaded or true otherwise.

boolean CU9000LoadResistSetup (String *path*)

Apply the given Resist Setup file to all active layers

Parameters:

path path to resist setup file

Returns:

false if the resist file could not be loaded or true otherwise.

boolean CU9000LoadResources (String *sPropertiesPath,*
String *sPropertiesName,*
String *sConversionFileName*
)

Loads given resource files

Parameters:

sPropertiesPath	properties directory
sPropertiesName	eg. DS_DI.properties file name
sConversionFileName	eg. odb2Li-Ledia.txt plotstamps conversion definition file

Returns:

`true` if the files have been correctly loaded; otherwise `false`

ObjectList CU9000OrderPlotstamps (Object[] *plotstamps,*
String *sLevel,*
String *sAtLevel,*
String *sStart,*
String *sOrder*
)

Reorder specific Plotstamps at given level.

Parameters:

plotstamps	the Object Array of the all plotstamps.
sLevel	the level of the plotstamps to be reordered (usually "pcb", "array").
sAtLevel	reordering may be related to given level (usually "panel" or "array").
sStart	may be one of "TL" - for top left, "TR" - top right, "BL" - bottom left, "BR" - bottom right
sOrder	may be one of "XX" - row ordered in the same orientation, "YY" - columns are ordered in the same direction, "XY" - rows are ordered zigzag, "YZ" - columns are ordered zigzag.

boolean CU9000Output (String *machine*)

Perform CU9000 output of active layers.

Parameters:

machine name of target machine

Returns:

false if a problem was detected during output or true otherwise

void CU9000SaveBPIs ( )

Saves BPI for current front layer and back layer if exists. Layer activity is also taken into account

void CU9000SaveLocalAlignmentPoints (String *sOutputFilePath*)

Saves local alignment points from current layer. The local alignment points are taken from BPI linked to

current layer

Parameters:

sOutputFilePath A full output file name. Eg. "T:\\CU9000job\\F_LOCAL_ALN_MARK.TXT"

**void CU9000SaveLocalAlignmentPoints (String *sOutputFilePath,*
boolean *bShareAlignmentMarks*
)**

Saves local alignment points from this layer data structure

Parameters:

sOutputFilePath A full output file name. Eg."T:\\CU9000job\\F_LOCAL_ALN_MARK.TXT"

bShareAlignmentMarks the alignment marks are shared with other areas if `true` and areas are set.

boolean CU9000SetParameters (String *xmlFile*)

Apply parameters for CU9000 from XML file to the active layers

Parameters:

xmlFile full path to XML file containing the settings to be applied. The XML file must conform to the DI_Settings schema

Returns:

false if the file was found to be invalid or if a problem occurred during applying the settings, or true otherwise

void cutToClipboard ( )

Delete all (selected) objects on layer in plane 1 Objects are stored in clipboard

**void dbBoolean (String *dbKey,*
Boolean *bValue*
)**

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:

dbKey modified or added Ucam.db key

bValue the value of the Ucam.db key.

boolean dbBoolean (String *dbKey*)

Returns boolean Ucam.db value of the given key

Parameters:

dbKey Ucam.db key

Returns:
boolean Ucam.db value of the given key or false if the key is not defined

**boolean dbBooleanDef (String *dbKey,*
boolean *bDefault*
)**

Returns boolean Ucam.db value of the given key

Parameters:
dbKey Ucam.db key
bDefault default value is returned if given key doesn't exist

Returns:
boolean Ucam.db value of the given key or false if the key is not defined

**void dbDouble (String *dbKey,*
Boolean *dValue*
)**

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:
dbKey modified or added Ucam.db key
dValue the value of the Ucam.db key.

double dbDouble (String *dbKey*)

Returns double Ucam.db value of the given key

Parameters:
dbKey Ucam.db key

Returns:
double Ucam.db value of the given key

**double dbDoubleDef (String *dbKey,*
double *dDefault*
)**

Returns double Ucam.db value of the given key

Parameters:
dbKey Ucam.db key
dDefault default value is returned if given key doesn't exist

Returns:
double Ucam.db value of the given key

void dbInteger (String *dbKey*,
Integer *iValue*
)

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:

dbKey modified or added Ucam.db key
iValue the value of the Ucam.db key.

int dbInteger (String *dbKey*)

Returns integer Ucam.db value of the given key

Parameters:

dbKey Ucam.db key

Returns:

integer Ucam.db value of the given key

int dbIntegerDef (String *dbKey*,
int *iDefault*
)

Returns integer Ucam.db value of the given key

Parameters:

dbKey Ucam.db key
iDefault default value is returned if given key doesn't exist

Returns:

integer Ucam.db value of the given key

void dbPath (String *dbKey*,
String *sPath*
)

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:

dbKey modified or added Ucam.db key
sPath the value of the Ucam.db key.

String dbPath (String *dbKey*)

Returns Path Ucam.db value of the given key

Parameters:

dbKey Ucam.db key

Returns:
Path Ucam.db value of the given key

String dbPathDef (String *dbKey*,
String *sDefault*
)

Returns Path Ucam.db value of the given key

Parameters:
dbKey Ucam.db key
sDefault default value is returned if given key doesn't exist

Returns:
Path Ucam.db value of the given key

void dbString (String *dbKey*,
String *sValue*
)

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:
dbKey modified or added Ucam.db key
sValue the value of the Ucam.db key.

String dbString (String *dbKey*)

Returns String Ucam.db value of the given key

Parameters:
dbKey Ucam.db key

Returns:
String Ucam.db value of the given key

String dbStringDef (String *dbKey*,
String *sDefault*
)

Returns String Ucam.db value of the given key

Parameters:
dbKey Ucam.db key
sDefault default value is returned if given key doesn't exist

Returns:
String Ucam.db value of the given key

void dbUnitValue (String *dbKey*,
String *sValue*
)

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:

dbKey modified or added Ucam.db key
sValue the value of the Ucam.db key (eg. "1 mil"...)

void dbUnitValue (String *dbKey*,
Double *dValue*
)

Modifies HOME Ucam.db file. Sets the value of the given key.

Parameters:

dbKey modified or added Ucam.db key
dValue the value of the Ucam.db key.

double dbUnitValue (String *dbKey*)

Returns Unit Ucam.db value of the given key

Parameters:

dbKey Ucam.db key

Returns:

Unit Ucam.db value of the given key

double dbUnitValueDef (String *dbKey*,
double *dDefault*
)

Returns Unit Ucam.db value of the given key

Parameters:

dbKey Ucam.db key
dDefault default value is returned if given key doesn't exist

Returns:

Unit Ucam.db value of the given key

double dbUnitValueDef (String *dbKey*,
String *sDefault*
)

Returns Unit Ucam.db value of the given key

Parameters:

dbKey Ucam.db key
sDefault default value is returned if given key doesn't exist

Returns:
Unit Ucam.db value of the given key

void defaultOrder ( )

Displays the current order of the rout path in a separate layer. Each chain of draws and arcs makes up a rout group which gets a sequence number. This number can be changed to change the rout order of the groups.

void defineFirst (double *p_x*,
double *p_y*
)

Defines start point of the chain. It is the from point of the closest element to defined point. The point is defined as X and Y world coordinates.

Parameters:
p_x (X coordinate) the new start poiunt
p_y (Y coordinate) the new start poiunt

void defineFirst (Point *p*)

Defines start point of the chain. It is the from point of the closest element to defined point. The point is defined as X and Y world coordinates.

Parameters:
p the new start poiunt

void defineGroup (double *p_x*,
double *p_y*,
int *iGroupNumber*
)

Defines group at given point with the given index. The point is defined as X and Y world coordinates.

Parameters:
p_x (X coordinate) the point
p_y (Y coordinate) the point
iGroupNumber the index of the defined group

void defineGroup (Point *p*,
int *iGroupNumber*
)

Defines group at given point with the given index. The point is defined as X and Y world coordinates.

Parameters:

p	the point
iGroupNumber	the index of the defined group

void defineSelectedGroup ( )

Defines group from selected elements.

void delete ( )

Delete all (selected) objects on active layers

See also:

deleteWithApe()

void deleteAllCFMEEAlignmentPoints ( )

Remove all alignment points

void deleteAllRefPoints (boolean *bOnAllActiveLay*)

Delete all reference points from layer

Parameters:

bOnAllActiveLay	if true it work on all active layers otherwise only on active loaded layer in plane 1

void deleteAllYsphotechAlignmentPoints (int *region*)

Remove all alignment point from a region

Parameters:

region	the regionnumber (0 for global)

void deleteAperture ( )

Aperture Manager: Deletes current Aperture

void deleteApertureAttribute (String *sAttributeName*)

Aperture Manager: Delete an Attribute of current Aperture

Parameters:

sAttributeName	The name of the Attribute to delete

void deleteCFMEEAlignmentPoint (int *point*)

Remove all alignment points

Parameters:

point the alignment point number

void deleteDouble ( )

Deletes overlapping parts of the draws. When two draws partially overlap the overlapping part of the draw is deleted. When two draws completely overlap the shortest draw is deleted.

**void deleteLayerByClass (String *className,*
String *subClass,*
int *num,*
String *side*
)**

Delete Layer by Class

Parameters:

className	The class of the layer wanted : "layer", "drill", "core" or "extra".
subClass	The subclass for the layer wanted. The default subclasses offered by Ucam are: "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".
num	The layer number. For "layer" and "drill" the layers are numbered from 1 to the number of layers for that class. For "extra" the layers are numbered per subclass.
side	Specifies the attachment for "extra" layers. Can be "top", "bottom", or "none"

void deleteLayersByActivation (boolean *active*)

Delete active/deactive layers

Parameters:

active Set true to delete activated and false to delete deactivated layers

void deleteLayersByName (String *layName*)

Delete Layers by name

Parameters:

layName Name (or name part) of any layer (layer, extra or drill)

See also:

deleteLayersByNames(String)

void deleteLayersByNames (String *layNames*)

Delete Layers by names **Example:**

- "test_txt" exact name
- "*_txt" or all layers with name ending "_txt"
- "tmp_*;*_tmp;_tmp_?" or semicolon separated list of wild cards

Parameters:

layNames semicolon separated list of the wild cards.

See also:

deleteLayersByName(String)

void deleteLayersByPlane (int *plane*)

Delete Layer by plane

Parameters:

plane Number of plane

void deleteRefPoint (int *iIndex*,
boolean *bOnAllActiveLay*
)

Delete refpoint from layer

Parameters:

iIndex The reference point number.
bOnAllActiveLay if true it work on all active layers otherwise only on active loaded layer in plane 1

void deleteRefPoints (ObjectList *Indexes*,
boolean *bOnAllActiveLay*
)

Delete reference points from layer

Parameters:

Indexes list of reference points numbers.
bOnAllActiveLay if true it work on all active layers otherwise only on active loaded layer in plane 1

void deleteSubJob (int *index*,
int *level*
)

Delete sub-job

Parameters:

index index - works on sub-job with the given index
level index - delete defined level of the sub-job on the index

void deleteTrueObjects ( )

deletes all True Objects contained in a given job

void deleteWithApe ( )

Delete all (selected) objects on active layers and remove unused apertures Not used by GUI

See also:
delete()

void deleteWorkspace (String *sWorkspaceName*)

Delete workspace layout file with a given name. NOTE: The same as menu command Workspaces > Delete... > Delete

Parameters:
sWorkspaceName

void deleteYsphotechAlignmentPoint (int *region*,
int *point*
)

Remove all alignment point from a region

Parameters:
region the regionnumber (0 for global)
point

void deleteZeroLengthDraws (double *dMaxLength*,
boolean *bFunctional*,
boolean *bNonFunctional*
)

Delete Zero Length Draws and Arcs

Parameters:
dMaxLength Maximum length of objects to be deleted
bFunctional Delete objects within functional copper if true
bNonFunctional Delete objects within non-functional copper if true

void deselectAll ( )

Deselect all objects.

void deselectAllApertures ( )

Aperture Manager: Deselect all Objects of all Apertures in Aperture list

void deselectAperture (ObjectList *apeIndexArray*)

Aperture Manager: Deselect Objects of Apertures

Parameters:

apeIndexArray Array of indexes of the apertures on the current layer

void deselectAperture ( )

Aperture Manager: Deselect Objects of current Aperture

**void deselectObjectAttribute (String *sAttrName,*
String *sAttrValue*
)**

Deprecated:

deselectObjectAttribute deselect objects with attribute with the given name and value from current job.

Parameters:

sAttrName The object attribute name
sAttrValue The object attribute value

void deselectObjectAttribute (String *sAttrName*)

Deprecated:

deselectObjectAttribute deselect objects with attribute with the given name from current job.

Parameters:

sAttrName The object attribute name

**void deselectObjectByAttribute (String *sAttrName,*
String *sAttrValue*
)**

deselectObjectAttribute deselect objects with attribute with the given name and value from current job.

Parameters:

sAttrName The object attribute name
sAttrValue The object attribute value

void deselectObjectByAttribute (String *sAttrName*)

deselectObjectAttribute deselect objects with attribute with the given name from current job.

Parameters:

sAttrName The object attribute name

**boolean detectPCBOutlines (String *sLayName,*
String *sParams*
)**

Detects PCB Outlines in current job

Parameters:

sLayName name of a new layer
sParams comma separated paremeters

Returns:

`true` if OK, otherwise `false` when something fails

**boolean DetectPlaceHolders (String *sHandlerName,*
String *sParams*
)**

The function detects all placeholders using customer's Handler.

Parameters:

sHandlerName Handler's name is the same as the name of registered Handler
sParams can be `null` or empty if it has to use default parameters. Given parameters are handled in specific Handler constructor.

Returns:

`true` when a layer with placeholder has been created and added to the Job, otherwise it returns `false`

**boolean DetectPlaceHoldersAtLayer (String *sHandlerName,*
String *sParams*
)**

The function detects all placeholders using customer's Handler.

Parameters:

sHandlerName Handler's name is the same as the name of registered Handler
sParams can be `null` or empty if it has to use default parameters. Given parameters are handled in specific Handler constructor.

Returns:

`true` when a layer with placeholder has been created and added to the Job, otherwise it returns `false`

**void dimensioning (String *sType,*
double *dApertureSize,*
ObjectList *oPoints,*
boolean *bShowErrors,*

```
                double      dArrowHeadWidth,
                double      dArrowHeadHeight,
                double      dRuleToElement,
                double      dRuleToDimLine,
                double      dTextToDimLine,
                double      dTextHeight,
                double      dTextWidth,
                double      dTolerancePos,
                double      dToleranceNeg,
                double      dToleranceScale,
                int         iFormat,
                boolean     bProjectionHorizontal,
                boolean     bProjectionVertical,
                String      sFontName,
                int         iFontStyle,
                int         iFontSize,
                String      sLabel
            )
```

Inserts an object of the given type in the layer using the list of points.

Parameters:

sType	dimensioning object type
dApertureSize	aperture size
oPoints	array of points
bShowErrors	When true, the errors are displayed, when false not. False is used for preview mode
dArrowHeadWidth	Width of the arrow head
dArrowHeadHeight	Height of the arrow head
dRuleToElement	The distance between the rule draws and the measurement points
dRuleToDimLine	The distance of the text to the text line
dTextToDimLine	The distance of the text to the text line
dTextHeight	The height of the text
dTextWidth	The width of the text. In case a font is defined, this value is ignored
dTolerancePos	The value for the positive tolerance. If both the positive and negative tolerance values are 0.0, no tolerance labels are displayed.
dToleranceNeg	The value for the negative tolerance. If both the positive and negative tolerance values are 0.0, no tolerance labels are displayed.
dToleranceScale	The fraction of the text height to use for the tolerance labels
iFormat	The number of decimals to use
bProjectionHorizontal	When true, the horizontal distance is shown. The angle distance is not shown when true
bProjectionVertical	When true, the vertical distance is shown. The angle distance is not shown when true
sFontName	font name, see java.awt.Font constructor
iFontStyle	font style, see java.awt.Font constructor
iFontSize	font size, see java.awt.Font constructor
sLabel	The text of Label

String direction ( )

Get the Move/Copy direction value

Returns:
"h" for horizontal, "v" for vertical or "" for free

void direction (String *sDirection*)

Set the Move/Copy direction value

Parameters:
sDirection "h" for horizontal, "v" for vertical or "" for free

void distance (double *distance*)

Set the Distance value

Parameters:
distance Value of the Distance

double distance ( )

Gets the Distance number value

Returns:
Value of the Distance

boolean distort (double *x*,
double *y*,
double *pCenter_x*,
double *pCenter_y*
)

Distort - distort the X and/or Y coordinates of the objects of a layer. Only the flash and draw coordinates are affected, not the pad sizes. For block apertures the data inside the block and the block flash points are distorted. The block options are not taken into account, so be careful with block options like rotation and mirror. Distort works on all active layers and on selected objects in a layer.

Parameters:
x - a multiplication value, X distort value
y - a multiplication value,Y distort value
pCenter_x (X coordinate) the center point will be used
pCenter_y (Y coordinate) the center point will be used

Returns:
true if distort has a problem, if no problem then returns false.

boolean distort (double *x*,
double *y*,
Point *pCenter*
)

Distort - distort the X and/or Y coordinates of the objects of a layer. Only the flash and draw coordinates are affected, not the pad sizes. For block apertures the data inside the block and the block flash points are distorted. The block options are not taken into account, so be careful with block options like rotation and mirror. Distort works on all active layers and on selected objects in a layer.

Parameters:

x - a multiplication value, X distort value
y - a multiplication value,Y distort value
pCenter the center point will be used

Returns:

true if distort has a problem, if no problem then returns false.

boolean distort (double *x,*
double *y*
)

Distort - distort the X and/or Y coordinates of the objects of a layer. Only the flash and draw coordinates are affected, not the pad sizes. For block apertures the data inside the block and the block flash points are distorted. The block options are not taken into account, so be careful with block options like rotation and mirror. Distort works on all active layers and on selected objects in a layer.

Parameters:

x - a multiplication value, X distort value
y - a multiplication value,Y distort value

Returns:

true if distort has a problem, if no problem then returns false.

void doActiveFunction ( )

DO function Execute current function using values entered in the Numbers **Example:**

```
setInPlane(1,1);
doMove(Point(100,200));
doActiveFunction();
doActiveFunction();
doActiveFunction();
doActiveFunction();
doCancelActiveFunction();
```

See also:

doMove(Upoint)

doCopy(Upoint)

doCancelActiveFunction()

void doCancelActiveFunction ( )

Cancel active function Resets current "Do" button function in the Numbers

See also:

doActiveFunction()

void doCopy (double *offset_x*,
double *offset_y*
)

Copy using parameters. doCopy respects current direction setting. **Example:**

```
setInPlane(1,1);
direction("");
copy(100,200);
doCopy(100,200);
doCopy(100,200);

direction("h");
copy(100,200);
doCopy(100,200);
doCopy(100,200);

direction("v");
copy(100,200);
doCopy(100,200);
doCopy(100,200);
```

Parameters:

offset_x (X coordinate) offset vector

See also:

copy(Upoint)

direction(String)

doActiveFunction()

doCancelActiveFunction()

Parameters:

offset_y (Y coordinate) offset vector

See also:

copy(Upoint)

direction(String)

doActiveFunction()

doCancelActiveFunction()

void doCopy (Point *offset*)

Copy using parameters. doCopy respects current direction setting. **Example:**

```
setInPlane(1,1);
direction("");
copy(100,200);
doCopy(100,200);
doCopy(100,200);

direction("h");
copy(100,200);
doCopy(100,200);
doCopy(100,200);

direction("v");
copy(100,200);
doCopy(100,200);
doCopy(100,200);
```

Parameters:

offset offset vector

See also:

copy(Upoint)

direction(String)

doActiveFunction()

doCancelActiveFunction()

void doMove (double *offset_x,*
double *offset_y*
)

Move using parameters. doMove respects current direction setting. **Example:**

```
setInPlane(1,1);
direction("");
move(100,200,false);
doMove(100,200);
doMove(100,200);

direction("h");
move(100,200,false);
doMove(100,200);
doMove(100,200);

direction("v");
move(100,200,false);
doMove(100,200);
doMove(100,200);
```

Parameters:

offset_x (X coordinate) offset vector

See also:

move(Upoint, boolean)

direction(String)

doActiveFunction()

doCancelActiveFunction()

Parameters:

offset_y (Y coordinate) offset vector

See also:

move(Upoint, boolean)

direction(String)

doActiveFunction()

doCancelActiveFunction()

void doMove (Point *offset*)

Move using parameters. doMove respects current direction setting. **Example:**

```
setInPlane(1,1);
direction("");
move(100,200,false);
doMove(100,200);
doMove(100,200);

direction("h");
move(100,200,false);
doMove(100,200);
doMove(100,200);

direction("v");
move(100,200,false);
doMove(100,200);
doMove(100,200);
```

Parameters:

offset offset vector

See also:

move(Upoint, boolean)

direction(String)

doActiveFunction()

doCancelActiveFunction()

---

void doOption (String *sOption*)

Sets the current operation mode

Parameters:

sOption - "sel" or "all" or "selall"

String doOption ( )

Gets the current operation mode

Returns:

mode ("sel" or "all" or "selall")

**void doRemoveAttribute (boolean *jobAttr,*
boolean *layAttr,*
boolean *apeAttr,*
boolean *objAttr*
)**

remove all attributes from job, layer, aperture, object

Parameters:

jobAttr if true remove attributes for job
layAttr if true remove attributes for layer
apeAttr if true remove attributes for aperture
objAttr if true remove attributes for object

```
void drag ( double  clickp_x,
            double  clickp_y,
            double  dRadius,
            double  offset_x,
            double  offset_y,
            double  rect_xmin,
            double  rect_ymin,
            double  rect_xmax,
            double  rect_ymax
          )
```

Drag Pad/track to new location

Parameters:

clickp_x	(X coordinate) Pad flashpoint / track endpoint
clickp_y	(Y coordinate) Pad flashpoint / track endpoint
dRadius	radius around click point where must be an object
offset_x	(X coordinate) Offset of new pad flashpoint / track endpoint
offset_y	(Y coordinate) Offset of new pad flashpoint / track endpoint
rect_xmin	(left boundary of rectangle) **Rectangle** of selection - all elements inside will be dragged
rect_ymin	(bottom boundary of rectangle) **Rectangle** of selection - all elements inside will be dragged
rect_xmax	(right boundary of rectangle) **Rectangle** of selection - all elements inside will be dragged
rect_ymax	(top boundary of rectangle) **Rectangle** of selection - all elements inside will be dragged

```
void drag ( Point      clickp,
            double     dRadius,
            Point      offset,
            Rectangle  rect
          )
```

Drag Pad/track to new location

Parameters:

clickp	Pad flashpoint / track endpoint
dRadius	radius around click point where must be an object
offset	Offset of new pad flashpoint / track endpoint
rect	**Rectangle** of selection - all elements inside will be dragged

```
void drag ( double  clickp_x,
            double  clickp_y,
            double  dRadius,
            double  offset_x,
            double  offset_y
          )
```

Drag Pad/track to new location

Parameters:

clickp_x	(X coordinate) Pad flashpoint / track endpoint
clickp_y	(Y coordinate) Pad flashpoint / track endpoint
dRadius	radius around click point where must be an object

offset_x (X coordinate) Offset of new pad flashpoint / track endpoint
offset_y (Y coordinate) Offset of new pad flashpoint / track endpoint

```
void drag ( Point   clickp,
            double  dRadius,
            Point   offset
          )
```

Drag Pad/track to new location

Parameters:

clickp Pad flashpoint / track endpoint
dRadius radius around click point where must be an object
offset Offset of new pad flashpoint / track endpoint

```
void dragAngle ( double   pt_x,
                 double   pt_y,
                 double   dRadius,
                 double   dist,
                 double   mlen,
                 boolean  bUseLimit
               )
```

Drag track angle corresponds to: Transform Objects - BGA Tracks - Drag Angle

Parameters:

pt_x (X coordinate) Start point of drag
pt_y (Y coordinate) Start point of drag
dRadius radius around click point where must be an object
dist Drag distance
mlen Minimum track length
bUseLimit use limit for move data

```
void dragAngle ( Point   pt,
                 double  dRadius,
                 double  dist,
                 double  mlen,
                 boolean bUseLimit
               )
```

Drag track angle corresponds to: Transform Objects - BGA Tracks - Drag Angle

Parameters:

pt Start point of drag
dRadius radius around click point where must be an object
dist Drag distance
mlen Minimum track length
bUseLimit use limit for move data

```
void dragAngle ( double  pt_x,
                 double  pt_y,
                 double  dRadius,
                 double  dist,
                 double  mlen
               )
```

Drag track angle corresponds to: Transform Objects - BGA Tracks - Drag Angle

Parameters:

pt_x	(X coordinate) Start point of drag
pt_y	(Y coordinate) Start point of drag
dRadius	radius around click point where must be an object
dist	Drag distance
mlen	Minimum track length

```
void dragAngle ( Point   pt,
                 double  dRadius,
                 double  dist,
                 double  mlen
               )
```

Drag track angle corresponds to: Transform Objects - BGA Tracks - Drag Angle

Parameters:

pt	Start point of drag
dRadius	radius around click point where must be an object
dist	Drag distance
mlen	Minimum track length

```
void dragLayer ( String   prevClass,
                 String   newClass,
                 int      prevPosition,
                 int      newPosition,
                 boolean  duplicate
               )
```

Drag Layer, optionally with duplicate

Parameters:

prevClass	"layer" or "extra" or "drill"
newClass	"layer" or "extra" or "drill"
prevPosition	index of the layer. For the "drill" layers it is from 1 to the number of drill layers. For the "layer" and "extra" it is from 1 to the number of all layers.
newPosition	index of the new layer. For the "drill" layers it is from 1 to the number of drill layers. For the "layer" and "extra" it is from 1 to the number of all layers.
duplicate	true makes exact copy of the original layer

Line dragLine (String *sLabel*)

Wait for user drag **Line**. It needs Ucam GUI. The function pauses script execution and waits for user interaction.

Parameters:

sLabel Label in information dialog

Returns:

Line dragged by user.

Exceptions:

AbortException after user abort

Rectangle dragRectangle (String *sLabel*)

Wait for user drag **Rectangle**. It needs Ucam GUI. The function pauses script execution and waits for user interaction.

Parameters:

sLabel Label in information dialog

Returns:

Rectangle dragged by user.

Exceptions:

AbortException after user abort

void drawLastPlanesInFront (boolean *bDo*)

Changes the drawing order of the planes. Planes 6-11 are placed in front of planes 1-5, while it is normally otherways.

Parameters:

bDo A flag indicating that this should happen, or should be undone.

void drawSlots (String *sDPFApeRef,*
double *dTolerance,*
String *sDPFSlotApe*
)

Replaces defined slots according to the given DPF aperture definition and tolerance with new aperture definition

Example:

```
setMode("selall", "mil", "no");
drawSlots("208=REC,110,90", 0.1, "215=CIR,90");
drawSlots("208=REC,110,90", 0.1, "215=CIR,90");
```

Parameters:

sDPFApeRef the DPF aperture definition - reference for search
dTolerance the tolerance for searching
sDPFSlotApe the DPF aperture definition - the new slot aperture

**void drawSlotsSelect (String *sDPFApe,*
double *dTolerance*
)**

Selects slots according to the given DPF aperture definition and tolerance

Example:

```
setMode("selall", "mil", "no");
selectSlots("208=REC,110,90", 0.1);
```

Parameters:

sDPFApe the DPF aperture definition
dTolerance the tolerance for searching

**void drillMapReplace (String *sSymbolFilePath,*
ObjectList *oMappingTable*
)**

method Replace from Drill Map dialog

Parameters:

sSymbolFilePath full path to Symbol dpf file
oMappingTable order of replaced apertures - see example

Example of order (2,0,1):

- the 1st drill aperture will be replaced by the 2nd aperture from symbols dpf file
- the 2nd drill aperture will not be replaced, index to symbols dpf file is 0 (zero)
- the 3rd drill aperture will be replaced by the 1st aperture from symbols dpf file

void DSAOIAlignmentApply ( )

Method simulates Alignment Apply button press

**void DSAOIAlignmentDetect (String *sMachineType,*
String *sObjectRestrictions,*
boolean *bPositive,*
boolean *bNegative,*
double *dMinimumSize,*
double *dMaximumSize*
)**

Method simulates Detect button press. Detects alignment points in the active layers

Parameters:

sMachineType The selected machine type used to determine the parameters
sObjectRestrictions A specification for the object restrictions
bPositive Allow positive objects as alignment points. Only used when a shape is defined

bNegative	Allow negative objects as alignment points. Only used when a shape is defined
dMinimumSize	The minimum size of the alignment point. (-1 means not active)
dMaximumSize	The maximum size of the alignment point. (-1 means not active)

boolean DSAOIApply ( )

Method simulates Apply button press

Returns:

true if the apply succeeded, false if not. The apply will not succeed if the fields Minimum **Line**, Minimum Space or Thickness are <= 0

```
void DSAOIAreaDetection ( boolean    detectImportantLine,
                          boolean    detectImportantSpace,
                          boolean    detectImportantClearance,
                          boolean    detectImportantDrill,
                          boolean    detectImportantFutureDrill,
                          boolean    detectImportantMaskOpenings,
                          boolean    detectProhibitCopper,
                          boolean    detectProhibitSpace,
                          boolean    detectProhibitNonfunctionalCopper,
                          ObjectList importantLineWidth,
                          Object[]   importantSpaceWidth,
                          double     importantClearanceWidth,
                          double     importantDrillSpreadValue,
                          double     prohibitLineWidth,
                          double     prohibitSpaceWidth,
                          double     minSliverSize,
                          String     pathStrategy,
                          boolean    mergeOutput,
                          boolean    clipProhibitWithImportant,
                          boolean    outputNormalAreas,
                          boolean    maskPolarity,
                          boolean    paintedArea,
                          double     paintedAreaValue,
                          String     PCBName,
                          boolean    outputDrillBinary
                        )
```

Auto-detect `important' and `prohibit' areas for the current job

Parameters:

detectImportantLine	whether or not to detect important areas in copper
detectImportantSpace	whether or not to detect important areas in space
detectImportantClearance	whether or not to detect clearance around drill holes
detectImportantDrill	whether or not to add actual drill holes to the important areas
detectImportantFutureDrill	whether or not to add future drill holes to the important areas
detectImportantMaskOpenings	whether or not to add solder mask openings as important areas
detectProhibitCopper	whether or not to detect prohibit areas in copper
detectProhibitSpace	whether or not to detect prohibit areas in space

detectProhibitNonfunctionalCopper	whether or not to add non-functional copper (NWP) as mask areas
importantLineWidth	maximum reported line width classes in important areas
importantSpaceWidth	maximum reported space width in important areas
importantClearanceWidth	maximum reported copper to drill hole clearance value in important areas
importantDrillSpreadValue	add this ring around the drill holes reported as important areas
prohibitLineWidth	maximum reported line width in prohibit areas
prohibitSpaceWidth	maximum reported space width in prohibit areas
minSliverSize	minimum size for slivers
pathStrategy	how to report functional copper where there are multiple paths ("thickest" or "all")
mergeOutput	whether or not to merge all important and all prohibit layers into just two layers
clipProhibitWithImportant	whether or not to clip the prohibit areas with the important areas
outputNormalAreas	whether or not to create a layer containing the areas which are neither important or prohibit
maskPolarity	if true, mask openings are positive; if false, resist is positive
paintedArea	create extra mask from the painted areas
paintedAreaValue	value used by painted area detection if allowed by "paintedArea"
PCBName	the name of the PCB block to use or null if none specified
outputDrillBinary	generate one binary file per copper layer with the drill information

```
void DSAOIAreaDetection ( boolean    detectImportantLine,
                          boolean    detectImportantSpace,
                          boolean    detectImportantClearance,
                          boolean    detectImportantDrill,
                          boolean    detectImportantFutureDrill,
                          boolean    detectImportantMaskOpenings,
                          boolean    detectProhibitCopper,
                          boolean    detectProhibitSpace,
                          boolean    detectProhibitNonfunctionalCopper,
                          ObjectList importantLineWidth,
                          Object[]   importantSpaceWidth,
                          double     importantClearanceWidth,
                          double     importantDrillSpreadValue,
                          double     prohibitLineWidth,
                          double     prohibitSpaceWidth,
                          double     minSliverSize,
                          String     pathStrategy,
                          boolean    mergeOutput,
                          boolean    clipProhibitWithImportant,
                          boolean    outputNormalAreas,
                          boolean    maskPolarity,
                          String     PCBName,
                          boolean    outputDrillBinary
                        )
```

Auto-detect `important' and `prohibit' areas for the current job

Parameters:

detectImportantLine	whether or not to detect important areas in copper
detectImportantSpace	whether or not to detect important areas in space

detectImportantClearance	whether or not to detect clearance around drill holes
detectImportantDrill	whether or not to add actual drill holes to the important areas
detectImportantFutureDrill	whether or not to add future drill holes to the important areas
detectImportantMaskOpenings	whether or not to add solder mask openings as important areas
detectProhibitCopper	whether or not to detect prohibit areas in copper
detectProhibitSpace	whether or not to detect prohibit areas in space
detectProhibitNonfunctionalCopper	whether or not to add non-functional copper (NWP) as mask areas
importantLineWidth	maximum reported line width classes in important areas
importantSpaceWidth	maximum reported space width in important areas
importantClearanceWidth	maximum reported copper to drill hole clearance value in important areas
importantDrillSpreadValue	add this ring around the drill holes reported as important areas
prohibitLineWidth	maximum reported line width in prohibit areas
prohibitSpaceWidth	maximum reported space width in prohibit areas
minSliverSize	minimum size for slivers
pathStrategy	how to report functional copper where there are multiple paths ("thickest" or "all")
mergeOutput	whether or not to merge all important and all prohibit layers into just two layers
clipProhibitWithImportant	whether or not to clip the prohibit areas with the important areas
outputNormalAreas	whether or not to create a layer containing the areas which are neither important or prohibit
maskPolarity	if true, mask openings are positive; if false, resist is positive
PCBName	the name of the PCB block to use or null if none specified
outputDrillBinary	generate one binary file per copper layer with the drill information

```
void DSAOIAreaDetection ( boolean    detectImportantLine,
                          boolean    detectImportantSpace,
                          boolean    detectImportantClearance,
                          boolean    detectImportantDrill,
                          boolean    detectImportantFutureDrill,
                          boolean    detectImportantMaskOpenings,
                          boolean    detectProhibitCopper,
                          boolean    detectProhibitSpace,
                          boolean    detectProhibitNonfunctionalCopper,
                          ObjectList importantLineWidth,
                          Object[]   importantSpaceWidth,
                          double     importantClearanceWidth,
                          double     importantDrillSpreadValue,
                          double     prohibitLineWidth,
                          double     prohibitSpaceWidth,
                          double     minSliverSize,
                          String     pathStrategy,
                          boolean    mergeOutput,
                          boolean    clipProhibitWithImportant,
                          boolean    outputNormalAreas,
                          boolean    maskPolarity,
                          String     PCBName
                        )
```

Auto-detect `important' and `prohibit' areas for the current job

Parameters:

detectImportantLine	whether or not to detect important areas in copper
detectImportantSpace	whether or not to detect important areas in space
detectImportantClearance	whether or not to detect clearance around drill holes
detectImportantDrill	whether or not to add actual drill holes to the important areas
detectImportantFutureDrill	whether or not to add future drill holes to the important areas
detectImportantMaskOpenings	whether or not to add solder mask openings as important areas
detectProhibitCopper	whether or not to detect prohibit areas in copper
detectProhibitSpace	whether or not to detect prohibit areas in space
detectProhibitNonfunctionalCopper	whether or not to add non-functional copper (NWP) as mask areas
importantLineWidth	maximum reported line width classes in important areas
importantSpaceWidth	maximum reported space width in important areas
importantClearanceWidth	maximum reported copper to drill hole clearance value in important areas
importantDrillSpreadValue	add this ring around the drill holes reported as important areas
prohibitLineWidth	maximum reported line width in prohibit areas
prohibitSpaceWidth	maximum reported space width in prohibit areas
minSliverSize	minimum size for slivers
pathStrategy	how to report functional copper where there are multiple paths ("thickest" or "all")
mergeOutput	whether or not to merge all important and all prohibit layers into just two layers
clipProhibitWithImportant	whether or not to clip the prohibit areas with the important areas
outputNormalAreas	whether or not to create a layer containing the areas which are neither important or prohibit
maskPolarity	if true, mask openings are positive; if false, resist is positive
PCBName	the name of the PCB block to use or null if none specified

```
void DSAOIAreaDetection ( boolean    detectImportantLine,
                          boolean    detectImportantSpace,
                          boolean    detectImportantClearance,
                          boolean    detectImportantDrill,
                          boolean    detectImportantFutureDrill,
                          boolean    detectImportantMaskOpenings,
                          boolean    detectProhibitCopper,
                          boolean    detectProhibitSpace,
                          boolean    detectProhibitNonfunctionalCopper,
                          ObjectList importantLineWidth,
                          double     importantSpaceWidth,
                          double     importantClearanceWidth,
                          double     importantDrillSpreadValue,
                          double     prohibitLineWidth,
                          double     prohibitSpaceWidth,
                          double     minSliverSize,
                          String     pathStrategy,
                          boolean    mergeOutput,
                          boolean    clipProhibitWithImportant,
                          boolean    outputNormalAreas,
                          boolean    maskPolarity,
                          String     PCBName
                        )
```

Auto-detect `important' and `prohibit' areas for the current job

Parameters:

detectImportantLine	whether or not to detect important areas in copper
detectImportantSpace	whether or not to detect important areas in space
detectImportantClearance	whether or not to detect clearance around drill holes
detectImportantDrill	whether or not to add actual drill holes to the important areas
detectImportantFutureDrill	whether or not to add future drill holes to the important areas
detectImportantMaskOpenings	whether or not to add solder mask openings as important areas
detectProhibitCopper	whether or not to detect prohibit areas in copper
detectProhibitSpace	whether or not to detect prohibit areas in space
detectProhibitNonfunctionalCopper	whether or not to add non-functional copper (NWP) as mask areas
importantLineWidth	maximum reported line width classes in important areas
importantSpaceWidth	maximum reported space width in important areas
importantClearanceWidth	maximum reported copper to drill hole clearance value in important areas
importantDrillSpreadValue	add this ring around the drill holes reported as important areas
prohibitLineWidth	maximum reported line width in prohibit areas
prohibitSpaceWidth	maximum reported space width in prohibit areas
minSliverSize	minimum size for slivers
pathStrategy	how to report functional copper where there are multiple paths ("thickest" or "all")
mergeOutput	whether or not to merge all important and all prohibit layers into just two layers
clipProhibitWithImportant	whether or not to clip the prohibit areas with the important areas
outputNormalAreas	whether or not to create a layer containing the areas which are neither important or prohibit
maskPolarity	if true, mask openings are positive; if false, resist is positive
PCBName	the name of the PCB block to use or null if none specified

void DSAOIAreasApply ( )

Method simulates Areas Apply button press

void DSAOIDetectRectangularArea (double *dMargin,*
String *sBlockMode,*
String *sPCBName,*
String *sReferenceLayerName,*
boolean *bIsSingleLevel,*
boolean *bIsInspection,*
boolean *bIsPmiP1,*
boolean *bIsPmiP2,*
boolean *bIsDrcP1,*
boolean *bIsDrcP2*
)

Automatically generate inspection areas for all active layers

Parameters:

dMargin	The found areas are spread with the given margin value
sBlockMode	Defines the reference layer for block detection. Possible values are

	BLOCK_OUTLINE, BLOCK_LAYER and BLOCK_REFERENCE_LAYER
sPCBName	The name of the PCB block to use or null if none specified
sReferenceLayerName	The name of the layer to be used if BLOCK_REFERENCE_LAYER mode is selected or null if none specified
bIsSingleLevel	When true, areas are generated for all active layers. When false, job level areas are generated
bIsInspection	When true, inspection areas are generated. When false, mask areas are generated
bIsPmiP1	Defines the PMI P1 setting for the generated areas
bIsPmiP2	Defines the PMI P2 setting for the generated areas
bIsDrcP1	Defines the DRC P1 setting for the generated areas
bIsDrcP2	Defines the DRC P2 setting for the generated areas

void DSAOILayerList ( )

Method simulates LayerList button press

void DSAOILayerListAreaApply ( )

Method simulates LayerList Area Apply button press

void DSAOILayerListAreaOutput ( )

Method simulates LayerList Area Output button press

void DSAOILayerListAreaSelect ( )

Method simulates LayerList Area Select button press

void DSAOILayerListGroupValue (String *sNewValue*)

Specifies Group in LayerList

Parameters:

sNewValue new value of the group

void DSAOILayerListRowDeselect (int *iIndexFrom*,
int *iIndexTo*
)

Method deselects rows in the LayerList between indexes (the first line has index 1)

Parameters:

iIndexFrom begin row index
iIndexTo end row index

void DSAOILayerListRowDeselect (int *iRow*)

Method deselects row in the LayerList (the first line has index 1)

Parameters:

iRow

void DSAOILayerListRowSelect (int *iIndexFrom*,
int *iIndexTo*
)

Method selects rows in the LayerList between indexes (the first line has index 1)

Parameters:

iIndexFrom begin row index
iIndexTo end row index

void DSAOILayerListRowSelect (int *iRow*)

Method selects row in the LayerList (the first line has index 1)

Parameters:

iRow

boolean DSAOILoadLayerListProfile (String *pro*)

Load a DS PI layer list profile

Parameters:

pro profile

Returns:

false if the profile was not loaded or true otherwise

boolean DSAOIOutput ( )

Perform DS PI output of the red layer and its back layer, if it is active

Returns:

false if a problem was detected during output or true otherwise

void DSAOIpinpointDetection (double *dStep*,
double *dInfinity*,
String *sOutputFilePath*
)

DS107 pinpoint detection

Parameters:

dStep	step between pinpoint locations
dInfinity	value bigger then specified are not important
sOutputFilePath	the complete file path including the file name

**void DSAOIpinpointDetection (double *dStep,*
double *dInfinity,*
String *sPCBName,*
String *sOutputFilePath,*
String *sLocation*
)**

DS107 pinpoint detection (in red and green layers)

Parameters:

dStep	step between pinpoint locations
dInfinity	value bigger then specified are not important
sPCBName	PCB name, can be "$" (complete layer), null or "" ("deepest level"), single PCB name or list of PCBs separated by ';'
sOutputFilePath	the complete file path including the file name
sLocation	

void DSAOIPositionApply ( )

Method simulates LayerList Area Apply button press

void DSAOISetApplyToBackLayers (boolean *bValue*)

sets value of the 'Apply To Back Layers' check box

Parameters:

bValue	value can be true (checked) or false (unchecked)

void DSAOISetApplyToFrontLayers (boolean *bValue*)

sets value of the 'Apply To Front Layers' check box

Parameters:

bValue	value can be true (checked) or false (unchecked)

**void DTMCalculate (String *sPlatingType,*
String *sToleranceScript*
)**

Calculates drill sizes according to given plating type and given tolerance VHS script

Parameters:

sPlatingType plating type, one from the Tooltable name
sToleranceScript tolerance script file name

boolean DTMCreateSymbolDrawing ( )

Creates layer with symbols and symbol table

Returns:

`true` if the Symbol drawings successfully created, otherwise `false`

boolean DTMLoadData ( )

Loads necessary data from drill layers and sets "uCustomerDiameter" attributes to holes

Returns:

boolean Activate the SDTMFrame UpdateDpfButton

void DTMLoadToleranceFile (String *sToleranceFileName*)

Loads tolerance file

Parameters:

sToleranceFileName Smart DrillTool Manager tolerance file name

void DTMRemoveAttributes ( )

Removes all attributes from drills

void DTMSaveDataToAttributes (String *sJobName*)

Saves the table data to attributes

Parameters:

sJobName

void DTMUpdateDPF (String *sJobName*)

Moves the plated objects to the plated layer and the unplated objects to the unplated layer

Parameters:

sJobName

void duplicateAperture ( )

Aperture Manager: Duplicates current Aperture

**void duplicateLayer (String *layName,*
String *newName*
)**

Duplicate Layer

Parameters:

layName Name of any layer (layer,extra or drill)
newName Name of duplicated layer

**void dwAnnotate (String *annotationLayerName,*
String *sChipID*
)**

The annotation layer describes positions of the DW Shots (chips) on panel. Each aperture in annotation layer has one flash point corresponding to one chip flash point in layer in plane 1. The aperture attribute dwShotAnnotation contains chip ID and position output in File A.

Parameters:

annotationLayerName the layer in job containing annotations
sChipID the chip defining block name

void dwAnnotate (String *annotationLayerName*)

The annotation layer describes positions of the DW Shots (chips) on panel. Each aperture in annotation layer has one flash point corresponding to one chip flash point in layer in plane 1. The aperture attribute dwShotAnnotation contains chip ID and position output in File A.

Parameters:

annotationLayerName the layer in job containing annotations

void dwApplyTransform (String *sResultFilePath*)

The Result file is the File B with real chip transformations. The method applies the transformations from the File B to layer in plane 1.

Parameters:

sResultFilePath full result File B path

**void editAperture (int *iApeNum,*
String *sApeName,*
String *sApeDef*
)**

Aperture Manager: Edit current Aperture

Parameters:

iApeNum	(new) Number of the Aperture
sApeName	(new) Name of the Aperture
sApeDef	(new) DPF style Definition String of Aperture, e.g. "REC,1.905,0.3048"

void emptyClipboard ( )

Empty clipboard

void enlargePads (String *absRel,*
double *aVal,*
boolean *bExclcon,*
boolean *bUseBGA*
)

Enlarge (stretch) pads with absolute/relative value corresponds to: Transform Objects - BGA Pads - Enlarge

Parameters:

absRel	Abs for absolute, Rel for relative
aVal	Enlarge value
bExclcon	If true, contours will be excluded
bUseBGA	If true, only objects with object attribute uBGA will be enlarged

String envString (String *name*)

Returns Environmental variable

Parameters:

name	e.g. "TEMP"

Returns:

the value of the given Environmental variable

void equalizeTrackSpace ( )

Equalize track space corresponds to: Transform Objects - BGA Tracks - Equalize Space

void etchCompensation (boolean *bUseExcludeAreas,*
boolean *bCreateLayerBackup,*
boolean *bShowLayerBackup,*
int *iOutStyle,*
boolean *bAsymmetricPadTrackComp,*
ObjectList *arrPrefOffset*
)

Etch Compensation without UseCompensateAreas -> reduced parameter set

Parameters:

bUseExcludeAreas	Use layer with 'Etch Exclusion' areas
bCreateLayerBackup	Copy all active layers for reference
bShowLayerBackup	Show reference layer, if existing
iOutStyle	0: default, 1: Output contour on top of original, 2: Output contour replace original
bAsymmetricPadTrackComp	Asymmetric pad track compensation
arrPrefOffset	parameter array for Preferred Offset: • String Offset Table • double Minimum Clearance • double for pads • double for tracks

```
void etchCompensation ( boolean    bUseExcludeAreas,
                        boolean    bUseCompensateAreas1,
                        boolean    bUseCompensateAreas2,
                        boolean    bCreateLayerBackup,
                        boolean    bShowLayerBackup,
                        int        iOutStyle,
                        boolean    bAsymmetricPadTrackComp,
                        ObjectList arrPrefOffset,
                        Object[]   arrCompLay1,
                        Object[]   arrCompLay2
                      )
```

Etch Compensation

Parameters:

bUseExcludeAreas	Use layer with 'Etch Exclusion' areas
bUseCompensateAreas1	Use compensation areas of lay 1 -> requires parameters in arrCompLay1
bUseCompensateAreas2	Use compensation areas of lay 2 -> requires parameters in arrCompLay2
bCreateLayerBackup	Copy all active layers for reference
bShowLayerBackup	Show reference layer, if existing
iOutStyle	0: default, 1: Output contour on top of original, 2: Output contour replace original
bAsymmetricPadTrackComp	Asymmetric pad track compensation
arrPrefOffset	parameter array for Preferred Offset: • String Offset Table • double Minimum Clearance • double for pads • double for tracks
arrCompLay1	parameter array for compensation areas of lay 1 • String Offset Table • double Minimum Clearance • double for pads • double for tracks
arrCompLay2	parameter array for compensation areas of lay 2 • String Offset Table • double Minimum Clearance

- double for pads
- double for tracks

void expandArcs ( )

expandArcs Replaces arcs by draws in a job. The arcs themselves are removed.

void expandBlock ( )

expandBlock Expands all data of block apertures in a job The block apertures themselves are removed from the aperture list.

void expandNibble (double *overlapValue,* double *pitchValue,* boolean *useOverlap*)

Expands elements to nibbles.

Parameters:

overlapValue	The overlap value of the nibbles
pitchValue	The pitch value of the nibbles
useOverlap	true if use the overlapValue, false if use the pitchValue

void expandText ( )

expandText Changes all text apertures in layers to flashes of each character with a complex aperture. This allows editing of 1 character and spread and choke of characters.

void expandTrueObjects ( )

expandTrueObjects Removes flashes, draws, arcs, regions that have a subobject and replaces them with their subobject.

void expandVtx ( )

expandVtx Expands all vector text data in a job.

void externalLinkManagerCheck ( )

The button Check on External Link Manager press

void filletJoin (double *pt_x*,
double *pt_y*,
double *dis*
)

Rounds the junction of two draws.

Parameters:

pt_x (X coordinate) The junction of the two draws
pt_y (Y coordinate) The junction of the two draws
dis Radius value of fillet (round)

void filletJoin (Point *pt*,
double *dis*
)

Rounds the junction of two draws.

Parameters:

pt The junction of the two draws
dis Radius value of fillet (round)

void fillPolygon (boolean *bDirection*)

Parameters:

bDirection direction

void fillPolygonCCW ( )

CCW fill polygon

void fillPolygonCW ( )

CW fill polygon

void fillWithAngledPattern (String *shape*,
double *size*,
double *pitch*,
double *angle*
)

Fills selected apertures with angled pattern

Parameters:

shape Dot shape ("circle", "square" and "diamond")
size Dot size

pitch Pitch between dots
angle Pattern angle

See also:

HyperShell::PATTERN_ANGLED_CIRCLE

HyperShell::PATTERN_ANGLED_SQUARE

HyperShell::PATTERN_ANGLED_DIAMOND

void fillWithPatternPads (String *sKind,*
boolean *bKeepEdge,*
double *pGridOrigin_x,*
double *pGridOrigin_y,*
double *pGridStep_x,*
double *pGridStep_y*
)

Replaces the selected contour with the choose pattern - fill with Pads

Parameters:

sKind odd - 1 active layer: the contours are filled with a pattern which alternates between the pads on the grid. The bottom left grid point remains contour free. - More than 1 active layers: the pattern alternates from top to bottom, starting with an Odd pattern for the upper active layer. Only active layers are considered. even - 1 active layer: the contours are filled with a pattern which alternates between the pads on the grid. The bottom left grid point of the contour is filled. - More than 1 active layers: the pattern alternates from top to bottom, starting with an Even pattern for the upper active layer. Only the active layers are considered. full - Patterns every layer.
bKeepEdge - if true keeps edge
pGridOrigin_x (X coordinate) - origin grid
pGridOrigin_y (Y coordinate) - origin grid
pGridStep_x (X coordinate) - step grid
pGridStep_y (Y coordinate) - step grid

void fillWithPatternPads (String *sKind,*
boolean *bKeepEdge,*
Point *pGridOrigin,*
Point *pGridStep*
)

Replaces the selected contour with the choose pattern - fill with Pads

Parameters:

sKind odd - 1 active layer: the contours are filled with a pattern which alternates between the pads on the grid. The bottom left grid point remains contour free. - More than 1 active layers: the pattern alternates from top to bottom, starting with an Odd pattern for the upper active layer. Only active layers are considered. even - 1 active layer: the contours are filled with a pattern which alternates between the pads on the grid. The bottom left grid point of the contour is filled. - More than 1 active layers: the pattern alternates from top to bottom, starting with an Even pattern for the upper active layer. Only the active layers are considered. full - Patterns every layer.
bKeepEdge - if true keeps edge
pGridOrigin - origin grid

pGridStep - step grid

void fillWithPatternStarburst (int *iSegments,*
String *sKind,*
int *dBlack,*
boolean *bWithCenter,*
double *pCenter_x,*
double *pCenter_y,*
boolean *bKeepEdge,*
double *dEdgeWith*
)

Replaces the selected contour with the choose pattern - fill the venting pattern with a starburst.

Parameters:

iSegments - The number of segments to be used.
sKind odd - Determines the start position of the segments. The first black segment lies on the positive X-axis. even - Determines the start position of the segments. The first white segment lies on the positive X-axis. alternate - Fills the selected layers alternately from top to bottom, starting with an even pattern for the upper active layer. Only the active layers are considered.
dBlack - Determines the relative portion of black and white areas for the filling pattern
bWithCenter - Defines the center point of the starburst.
pCenter_x (X coordinate) - the center point
pCenter_y (Y coordinate) - the center point
bKeepEdge - if true keeps edge
dEdgeWith - edge with

void fillWithPatternStarburst (int *iSegments,*
String *sKind,*
int *dBlack,*
boolean *bWithCenter,*
Point *pCenter,*
boolean *bKeepEdge,*
double *dEdgeWith*
)

Replaces the selected contour with the choose pattern - fill the venting pattern with a starburst.

Parameters:

iSegments - The number of segments to be used.
sKind odd - Determines the start position of the segments. The first black segment lies on the positive X-axis. even - Determines the start position of the segments. The first white segment lies on the positive X-axis. alternate - Fills the selected layers alternately from top to bottom, starting with an even pattern for the upper active layer. Only the active layers are considered.
dBlack - Determines the relative portion of black and white areas for the filling pattern
bWithCenter - Defines the center point of the starburst.
pCenter - the center point
bKeepEdge - if true keeps edge
dEdgeWith - edge with

void fillWithPatternTracks (String *sPattern,*
double *dStep,*
double *dWidth,*
double *dRotation,*
boolean *bKeepEdge*
)

Replaces the selected contour with the choose pattern - fill the venting pattern with tracks

Parameters:

sPattern - Choose the required pattern
dStep - Defines the step between 2 lines.
dWidth - Defines the width of the lines (width < step).
dRotation - Defines the rotation for the pattern (CCW).
bKeepEdge - if true keeps edge

int fillWithVectors (double *dOverlap,*
double *dDiameter,*
int *iApeCount,*
int *iApeNum,*
String *sFillOpt*
)

Fills the specified apertures with circular draws.

Parameters:

dOverlap Specifies the overlap for the apertures.
dDiameter Specifies the diameter of the fill apertures. The first aperture has a diameter dDiameter. The n-th has a diameter n*dDiameter.
iApeCount The number of fill apertures.
iApeNum The number of the first fill aperture.
sFillOpt Specifies which kind of apertures should be filled up. This can be any combination of "com", "box", "con" and "txt". Boxes, complexes and text apertures are replaced by a block aperture.

Returns:

number of errors

void findSections (String *szOptions*)

Find job sections

Parameters:

szOptions options ("sel" or "all" or "selall")

void findSlots ( )

Find "IPCATG" slots according keys that defines them.

int findStandardShape (double *dTolerance,*
String *szOpt,*
String *szAction*
)

Find standard shapes within COM and CON apertures on layer

Parameters:

dTolerance	tolerance
szOpt	options "con,com,blo:cir,rec,box,the" con, com or blo means what to consider (blo means go into block definitions), cir and rec means what to replace with, you can also use * as joker -> "*:rec,cir, box" or "*:*" or "con,com:*"
szAction	("select", "replace"), in case of select all regions and/or complexes will be assigned attribute "uStandardShape" with value standard shape defintion

Returns:

amount of find/replaced apertures, 0 when no, -1 on error

void flashMakerDeleteComplex ( )

Deletes all complexes from founded list of painted objects. FlashMaker Delete Complex

void flashMakerDeselectComplex ( )

Deselects all complexes from founded list of painted objects. FlashMaker Deselect Complex

void flashMakerFind ( )

Finds all painted objects that are suitable to replace with flashes. FlashMaker Find button

void flashMakerFindStandardShapes (Uxjob *oJob*)

Finds all REGs and COMs that are suitable to replace with flashes of standard shapes (CIR and REC). FlashMaker Find button for Con&Com tab

Parameters:

oJob current job

void flashMakerReplace ( )

Replaces all found pianted objects with flashes FlashMaker Replace button

void flashMakerReplaceStandardShapes (Uxjob *oJob*)

Replaces all found pianted objects with flashes FlashMaker Replace button

Parameters:

oJob current job

```
void flashMakerSetup ( double  minCutoff,
                       double  minSize,
                       double  maxSize,
                       boolean useTol,
                       double  tol,
                       boolean useMask,
                       boolean deselNonModel
                     )
```

FlashMaker setup

Parameters:

minCutoff	Minimum rounding value for box corners If smaller rounding is found, box will be replaced by rectangle
minSize	Minimum x/y size of pads to be replaced. (0 = replace all)
maxSize	Maximum x/y size of pads to be replaced
useTol	Use a tolerance value (if false, optimal tolerance is used)
tol	Tolerance on size of replaced objects
useMask	Only look at pads free of soldermask
deselNonModel	Finds pads in selected data. Deselects all other objects.

```
void flashMakerSetup ( double  minCutoff,
                       double  minSize,
                       double  maxSize,
                       boolean useTol,
                       double  tol,
                       boolean useMask,
                       boolean completelyFree,
                       boolean deselNonModel
                     )
```

FlashMaker setup

Parameters:

minCutoff	Minimum rounding value for box corners If smaller rounding is found, box will be replaced by rectangle
minSize	Minimum x/y size of pads to be replaced. (0 = replace all)
maxSize	Maximum x/y size of pads to be replaced
useTol	Use a tolerance value (if false, optimal tolerance is used)
tol	Tolerance on size of replaced objects
useMask	Only look at pads free of soldermask
completelyFree	Only look at pads completely free of mask
deselNonModel	Finds pads in selected data. Deselects all other objects.

void flipJob (String *mirror,*
boolean *bFlipBuildup,*
boolean *bFlipAttachNone,*
String *suffix*
)

Flip the job buildup

Parameters:

mirror	Either "x", or "y" to mirror the layers.
bFlipBuildup	Either false, or true to flip the job buildup.
bFlipAttachNone	Either false, or true to flip layers of class "extra" and attach "none".
suffix	A suffix that is added to the layer names.

- Flip adds the 'uFlipJob' attribute to the job. It's value is 'x', 'y', 'xy' or 'yx'. Flipping toggles the attribute, i.e. flip x + flip x, results in uFlipJob = ''.
- The suffix is toggled, so no _x_x when flipped twice, but the original layer name again.
- PlotParameter Mirror is swapped
- Readable Side is swapped

void forEachApe (String *sType*)

Do the operations on all apertures of the given type.

Parameters:

sType it must be on from the following

- "cir", "rec", "box", "oct", "con", "com", "the", "txt", "blo", "squ" and "don".

Example:

```
print("JOB NAME " + jobName());
forEachLayer() {
    print("\n--- Layer " + layName() + " ---");

    forEachApe("cir") {
        String shape = apeShape();

        print("  Aperture " + apeInfo());
        print("    outer: " + apeOuter());
    }
}
```

void forEachApe ( )

Do the operations on all apertures. **Example:**

```
print("JOB NAME " + jobName());
forEachLayer() {
    print("\n--- Layer " + layName() + " ---");

    forEachApe() {
        String shape = apeShape();

        print("  Aperture " + apeInfo());
        if (isEqual(shape, "cir") || isEqual(shape, "don")  || isEqual(shape,
"the")) {
            print("  outer: " + apeOuter());
            if (!isEqual(shape, "cir")) {
                print("  inner: " + apeInner());
            }
```

```
        }
        else if (isEqual(shape, "rec") || isEqual(shape, "box")) {
            print("  X size: " + apeXSize());
            print("  Y size: " + apeYSize());
        }
    }
}
```

void forEachArc ( )

Do the operations on all arcs of the current aperture.

void forEachDraw ( )

Do the operations on all draws of the current aperture.

void forEachDrill (String *sSubClass*)

Do the operations on all drill layers of the given subclass.

Parameters:

sSubClass The subclass for the drill layer wanted. If not important specify "". The default subclasses offered by Ucam are "drill", "buried", "blind", "plated", "unplated" and "fixing".

void forEachDrill ( )

Do the operations on all drill layers. **Example:**

```
print("--- JOB statistics ---");
print("Name: " + jobName());
print("Spec: " + jobSpec());
print("Path: " + jobPath());
print("Customer: " + jobCustomer());
print("Revision: " + jobRevision());
int layerCount = 1;
forEachDrill() {
    print("\n  --- Layer " + layerCount++ + " ---");
    print("    Name: " + layName());
    print("    Class: " + layClass());
    print("    Subclass: " + laySubClass());
    print("    Attach: " + layAttach());
    print("    Readable: " + layReadable());
    print("    Reverse: " + layReverse());
}
```

void forEachExtra (String *sSubClass,* String *sAttach*)

Do the operations on all extra layers of the given subclass and attachment.

Parameters:

sSubClass The subclass for the layer wanted. If not important specify "". The default subclasses offered by Ucam are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

sAttach specifies the attachment for "extra" layers. Can be "top", "bottom", "none" or "all".

void forEachExtra (String *sSubClass*)

Do the operations on all Extra layers of the given subclass.

Parameters:

sSubClass The subclass for the layer wanted. If not important specify "". The default subclasses offered by Ucam are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate" for extra layers. "drill", "buried", "blind", "plated", "unplated" and "fixing" for drill layers. "outer", "inner" and "mixed" for signal layers.

void forEachExtra ( )

Do the operations on all extra layers. **Example:**

```
print("--- JOB statistics ---");
print("Name: " + jobName());
print("Spec: " + jobSpec());
print("Path: " + jobPath());
print("Customer: " + jobCustomer());
print("Revision: " + jobRevision());
int layerCount = 1;
forEachExtra() {
    print("\n  --- Layer " + layerCount++ + " ---");
    print("    Name: " + layName());
    print("    Class: " + layClass());
    print("    Subclass: " + laySubClass());
    print("    Attach: " + layAttach());
    print("    Readable: " + layReadable());
    print("    Reverse: " + layReverse());
}
```

void forEachFlash ( )

Do the operations on all flashes of the current aperture.

void forEachI8Job (String *serverName,*
String *dbname,*
String *username,*
String *password,*
String *context,*
String *i8path*
)

Do the operations on all jobs in the given database context.

Parameters:

serverName * name of the database server
dbname * name of the database to use (usually 'integr8tor')
username * database login
password * database password
context * integr8tor context to use (usually 'autoflow')
i8path * path to your integr8tor installation (parent of the Integr8tor folder)

**void forEachInRectangle (Rectangle *rect,*
boolean *opt*
)**

Do the operation on the objects whose enclosing rectangle overlap the specified rectangle.

Parameters:

rect The target rectangle.

opt if true, all the objects which actually overlap the rectangle are found, otherwise all objects whose enclosing rectangle overlaps the specified rectangle are found.

void forEachInRectangle (Rectangle *rect*)

Do the operation on the objects which actually overlap the specified rectangle.

Parameters:

rect The target rectangle.

Object forEachItem (ObjectList *items*)

Do the operation on the all items in given Object List.

Example:

```
int count = 0;
fileName = forEachItem(osGetFileList(chooseDirPath())) {
    print(fileName);
    count++;
}
print(count + " file(s) in directory.");
```

Parameters:

items Object List its items will be iterated.

Returns:

returns each object from given ObjectList in loop

See also:

osGetFileList(String)

osGetFileList(String, boolean)

osGetFileList(String, String, boolean)

chooseDirPath()

void forEachJobNet ( )

Create the net iterator over the all job layers

**void forEachLayer (String *sClass,*
String *sSubClass,***

String *sAttach*
)

Do the operations on all layers of the given class, subclass and attachment.

Parameters:

sClass "layer" or "drill" or "extra"

sSubClass The subclass for the layer wanted. If not important specify "". The default subclasses offered by Ucam are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

sAttach specifies the attachment for "extra" layers. Can be "top", "bottom", "none" or "all".

void forEachLayer (String *sClass,*
String *sSubClass*
)

Do the operations on all layers of the given class and subclass.

Parameters:

sClass "layer" or "drill" or "extra"

sSubClass The subclass for the layer wanted. If not important specify "". The default subclasses offered by Ucam are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate" for extra layers. "drill", "buried", "blind", "plated", "unplated" and "fixing" for drill layers. "outer", "inner" and "mixed" for signal layers.

void forEachLayer (String *sClass*)

Do the operations on all layer of the given class.

Parameters:

sClass it must be

- "layer" (means signal layers)
- "drill"
- "extra"

void forEachLayer ( )

Do the operations on all layers. **Example:**

```
print("--- JOB statistics ---");
print("Name: " + jobName());
print("Spec: " + jobSpec());
print("Path: " + jobPath());
print("Customer: " + jobCustomer());
print("Revision: " + jobRevision());
int layerCount = 1;
forEachLayer() {
    print("\n  --- Layer " + layerCount++ + " ---");
    print("    Name: " + layName());
    print("    Class: " + layClass());
    print("    Subclass: " + laySubClass());
    print("    Attach: " + layAttach());
    print("    Readable: " + layReadable());
    print("    Reverse: " + layReverse());
}
```

void forEachLayerNet ( )

Create the net iterator over the layer in plane 1

void forEachNet (int *iNet*)

Do the operations on all objects with the given net number on current layer. It goes over the all apertures in current layer. Example:

```
int objCount = 0;
int net = 20;
forEachNet(net) {
    objSelect("+");
    objCount++;
}
print(objCount + " object(s) with the net number " + net + " selected.");
```

Parameters:

iNet The net number we look for

void forEachObject (String *sClass*)

Do the operations on all objects of the given class of the current aperture.

Parameters:

sClass it must be on from the following

- Possible values are "arc", "dra", "fla" or "vtx".

void forEachObject ( )

Do the operations on all objects of the current aperture. **Example:**

```
print("JOB NAME " + jobName());
forEachLayer() {
    print("\n--- Layer " + layName() + " ---");
    forEachApe("cir") {
        print(" Aperture " + apeInfo());
        forEachObject() {
            print("    " + objInfo());
        }
    }
}
```

void forEachPEInputJob ( )

Do the operations on all PanelEditor input jobs of the current PanelJob.

void forEachPEPanelJob ( )

Do the operations on all proposed PanelEditor PanelJob of the current solution.

void forEachPESolution ( )

Do the operations on all proposed PanelEditor solutions.

void forEachRegion ( )

Do the operation on the all regions of the Contours.

void forEachSignal (String *sSubClass*)

Do the operations on all signal layers of the given subclass.

Parameters:

sSubClass The subclass for the signal layer wanted. If not important specify "". The default subclasses offered by Ucam are "outer", "inner" and "mixed".

void forEachSignal ( )

Do the operations on all signal layers. **Example:**

```
print("--- JOB statistics ---");
print("Name: " + jobName());
print("Spec: " + jobSpec());
print("Path: " + jobPath());
print("Customer: " + jobCustomer());
print("Revision: " + jobRevision());
int layerCount = 1;
forEachSignal() {
    print("\n  --- Layer " + layerCount++ + " ---");
    print("    Name: " + layName());
    print("    Class: " + layClass());
    print("    Subclass: " + laySubClass());
    print("    Attach: " + layAttach());
    print("    Readable: " + layReadable());
    print("    Reverse: " + layReverse());
}
```

void forEachVtxt ( )

Do the operations on all vector texts of the current aperture.

int GDSII_outLayer (String *filename*)

Generate GDSII from the current layer.

See also:

GDSII_outLayer(String, String)

GDSII_outLayer(String, String)

Parameters:

filename

**int GDSII_outLayer (String *filename,*
String *options*
)**

Generate GDSII from the current layer.

Parameters:

filename ... full specification of the GDSII file to be generated

options ... set of options separated by comma (can be empty): 'ARCEXPAND=units' ... provide precision for the expansion of arcs (like ARCEXPAND=0.01mm) 'DPF' ... in addition to the GDSII file, generate corresponding DPF file too (for verification/debugging purposes) 'PRECISE' ... use all post-merge steps in final contourization (typically substantially slower) 'PIECEWISE' ... flatten cells and write them by pieces instead of providing one big contour for each cell(keep array structure) 'NODATE' ... use the same date (1.1.1970 0:00.00) instead of the current date/time) for all items in the GDSII file 'FEEDBACK' ... show data in various steps of the process in feedback layers (for verification/debugging purposes)

Returns:

status (0 if everything was OK)

**boolean generateContours (double *dGap,*
double *dOverlap*
)**

Generate a Contour (Close)

Parameters:

dGap Maximum opening that can be closed

dOverlap Maximum overlap that can be deleted

Returns:

status

**boolean generateContoursOnLayer (double *dGap,*
double *dOverlap*
)**

Generate a Contour (Close)

Parameters:

dGap Maximum opening that can be closed

dOverlap Maximum overlap that can be deleted

Returns:

status

ObjectList getAttrCategories ( )

Returns ObjectList of the all available attribute categories

Returns:

ObjectList of the all available attribute categories

ObjectList getAttrNames (String *sCategory*)

Return ObjectList of the attribute names in given category

Parameters:

sCategory Attribute category name

See also:

getAttrCategories()

Returns:

ObjectList of the attribute names in given category

ObjectList getAttrValues (String *sAttributeName*)

Returns ObjectList of the available values for attribute with given name

Parameters:

sAttributeName Attribute name

Returns:

ObjectList of the available values for attribute with given name

int getCount (String *sType*)

Returns number of the faults of the given type

Parameters:

sType fault type name

Returns:

Number of the faults of the given type

ObjectList getFaultTypes ( )

Returns fault type names in ObjectList

Returns:

ObjectList of the fault type names

String getFileLastModified (ObjectList *fileInfo*)

Returns the time that the file denoted by this fileInfo was last modified.

Parameters:

fileInfo objectlist with the file information

Returns:

a string representation of the last modification date.

See also:
HSH_base::osFileInfo(String)

String getFileName (ObjectList *fileInfo*)

Returns the name of the file or directory denoted by this fileInfo.

Parameters:
fileInfo objectlist with the file information

Returns:
the name of the file or directory denoted by this fileInfo

See also:
HSH_base::osFileInfo(String)

String getFileParent (ObjectList *fileInfo*)

Returns the pathname string of this fileInfo's parent, or null if this fileInfo does not name a parent directory.

Parameters:
fileInfo objectlist with the file information

Returns:
the pathname string of this fileInfo's parent, or null if this fileInfo does not name a parent directory

See also:
HSH_base::osFileInfo(String)

long getFileSize (ObjectList *fileInfo*)

Returns the length of the file denoted by this fileInfo. The return value is unspecified if this fileInfo does not denotes a file.

Parameters:
fileInfo objectlist with the file information

Returns:
the length, in bytes, of the file denoted by this fileInfo, or 0 otherwise.

See also:
HSH_base::osFileInfo(String)

Ulayer getLayer (ObjectList *layerID*)

Get Layer by the given ID

Returns:
layer matching to the given ID; or `null`

Parameters:

layerID

ObjectList getLayerNames ( )

Returns ObjectList containing layer names in ObjectList

Warning: The Object list may contain the same name more then once in case the job has not unique Layer names.

Example: Prints out all layers' name.

```
int counter = 1;
item = forEachItem(getLayerNames()) {
item = forEachItem(getLayerNames()) {
    print("Layer name " + (counter++) + " is " + item);
}
```

Returns:

ObjectList with all layer names

ObjectList getLayers ( )

Returns structured information about all layers in ObjectList Items in ObjectList are ObjectLists with layer information in fixed order.

Note: The function is used to get information about layers in job without changes in planes and layer activities

Item structure:

```
info[LAYER_NAME] - Layer name
info[LAYER_CLASS] - Layer class
info[LAYER_SUBCLASS] - Layer subclass
info[LAYER_ATTACH] - Layer attachment
info[LAYER_INDEX] - Layer index
info[LAYER_ACTIVITY] - Layer activity
info[LAYER_APERTURES] - Layer number of apertures or -1 if the layer is not loaded yet.
```

Example: Prints out all layers' info.

```
int counter = 1;
info = forEachItem(getLayers()) {
info = forEachItem(getLayers()) {
    print("---- Layer " + (counter++) + " ----");
    print("  name " + info[LAYER_NAME]);
    print("  class " + info[LAYER_CLASS]);
    print("  subclass " + info[LAYER_SUBCLASS]);
    print("  attach " + info[LAYER_ATTACH]);
    print("  index " + info[LAYER_INDEX]);
    print("  active " + info[LAYER_ACTIVITY]);
    print("  apertures " + info[LAYER_APERTURES]);
}
```

Returns:

ObjectList with layers' info (ObjectList)

See also:

LAYER_NAME

LAYER_CLASS

LAYER_SUBCLASS

LAYER_ATTACH

LAYER_INDEX

LAYER_ACTIVITY

LAYER_APERTURES

Rectangle getLocationOnScreen (String *sFrameName*)

Gives screen location and dimension of this dockable frame.

Parameters:

sFrameName identification frame given by getFrameID() method of CustomFrame class

Returns:

location and width and height of the dockable frame as an **Rectangle**

ObjectList getMode ( )

Gets the current operation mode, units and snapmode

Returns:

array of [{sOption, sUnit, sSnapMode}]

See also:

com.barco.ets.ucam.hypershell.HyperShell::setMode(String, String, String)

ObjectList getNetAttrNames ( )

Return ObjectList of the net attribute names in given Layer

Returns:

ObjectList of the net attribute names

ObjectList getNetNames ( )

Return ObjectList of the net names in current job

Returns:

ObjectList of the net names

int getNetNumberByClick (double *pt_x,*
double *pt_y*
)

Found object in layer on plane 1 on the coordinates and returns its net number

Parameters:

pt_x (X coordinate) coordinates, click point

pt_y (Y coordinate) coordinates, click point

Returns:

net number a object on the coordinate or 0: no net or -1: no object on the coordinate

int getNetNumberByClick (Point *pt*)

Found object in layer on plane 1 on the coordinates and returns its net number

Parameters:

pt coordinates, click point

Returns:

net number a object on the coordinate or 0: no net or -1: no object on the coordinate

Ulayer getNextLayer ( )

Returns next layer after current layer in plane 1

Returns:

next layer after current layer in plane 1

ObjectList getODBxxSteps (String *sPath*)

getODBxxSteps

Parameters:

sPath path to ODB job

Returns:

two item ObjectArray, the first item is Object List of step names and the second item is Object List of layer names.

int getPlotParam (String *sKey*,
int *iDefValue*
)

Sets the plot parameter.

Parameters:

sKey The key. Possible values are :

- "RESOLUTION"
- "XOFF" (the x offset)
- "YOFF" (the y offset)
- "XSCALE" (the x scale)
- "YSCALE" (the y scale)
- "SXCEN" (the x scale center)
- "SYCEN" (the y scale center)
- "MXCEN" (the x mirror center)
- "MYCEN" (the y mirror center)
- "POSITION":
- - MergeJob.PLOT_POSITION_COMBINE_FILL_PERCENTAGE
- - MergeJob.PLOT_POSITION_COMBINE
- - MergeJob.PLOT_POSITION_SINGLE
- - MergeJob.PLOT_POSITION_SINGLE_LEFT_FIXED
- - MergeJob.PLOT_POSITION_SINGLE_RIGHT_FIXED

- "ENLARGE_VECTORS_MINIMUMSIZE"
- "ENLARGE_VECTORS_AMOUNT"
- "ENLARGE_FLASH_MINIMUMSIZE"
- "ENLARGE_FLASH_AMOUNT"
- "ENLARGE_CONDUCTOR_SIZE" (Adds a size. Use 0 to remove them all.)
- "ENLARGE_CONDUCTOR_AMOUNT"
- "ENLARGE_COMPLEX"
- "APPLY_ENLARGE_TO" (The value must be one of Ulayer.PLOT_ENLARGE_*)
- "VARIABLE_TEXT_HEIGHT"
- "VARIABLE_TEXT_DIRECTION" (The value must be one of Ulayer.PLOT_VARTEXT_DIRECTION_*)

iDefValue The default value

Returns:
value of the plot parameter or default value

**double getPlotParam (String *sKey,*
double *dDefValue*
)**

Sets the plot parameter.

Parameters:
sKey The key. Possible values are :

- "RESOLUTION"
- "XOFF" (the x offset)
- "YOFF" (the y offset)
- "XSCALE" (the x scale)
- "YSCALE" (the y scale)
- "SXCEN" (the x scale center)
- "SYCEN" (the y scale center)
- "MXCEN" (the x mirror center)
- "MYCEN" (the y mirror center)
- "POSITION":
- - MergeJob.PLOT_POSITION_COMBINE_FILL_PERCENTAGE
- - MergeJob.PLOT_POSITION_COMBINE
- - MergeJob.PLOT_POSITION_SINGLE
- - MergeJob.PLOT_POSITION_SINGLE_LEFT_FIXED
- - MergeJob.PLOT_POSITION_SINGLE_RIGHT_FIXED
- "ENLARGE_VECTORS_MINIMUMSIZE"
- "ENLARGE_VECTORS_AMOUNT"
- "ENLARGE_FLASH_MINIMUMSIZE"
- "ENLARGE_FLASH_AMOUNT"
- "ENLARGE_CONDUCTOR_SIZE" (Adds a size. Use 0 to remove them all.)
- "ENLARGE_CONDUCTOR_AMOUNT"
- "ENLARGE_COMPLEX"
- "APPLY_ENLARGE_TO" (The value must be one of Ulayer.PLOT_ENLARGE_*)
- "VARIABLE_TEXT_HEIGHT"
- "VARIABLE_TEXT_DIRECTION" (The value must be one of Ulayer.PLOT_VARTEXT_DIRECTION_*)

dDefValue The default value

Returns:
value of the plot parameter or default value

String getPlotParam (String *sKey*,
String *sDefValue*
)

Sets the plot parameter.

Parameters:

sKey The key. Possible values are :

- "RESOLUTION"
- "XOFF" (the x offset)
- "YOFF" (the y offset)
- "XSCALE" (the x scale)
- "YSCALE" (the y scale)
- "SXCEN" (the x scale center)
- "SYCEN" (the y scale center)
- "MXCEN" (the x mirror center)
- "MYCEN" (the y mirror center)
- "POSITION":
- - MergeJob.PLOT_POSITION_COMBINE_FILL_PERCENTAGE
- - MergeJob.PLOT_POSITION_COMBINE
- - MergeJob.PLOT_POSITION_SINGLE
- - MergeJob.PLOT_POSITION_SINGLE_LEFT_FIXED
- - MergeJob.PLOT_POSITION_SINGLE_RIGHT_FIXED
- "ENLARGE_VECTORS_MINIMUMSIZE"
- "ENLARGE_VECTORS_AMOUNT"
- "ENLARGE_FLASH_MINIMUMSIZE"
- "ENLARGE_FLASH_AMOUNT"
- "ENLARGE_CONDUCTOR_SIZE" (Adds a size. Use 0 to remove them all.)
- "ENLARGE_CONDUCTOR_AMOUNT"
- "ENLARGE_COMPLEX"
- "APPLY_ENLARGE_TO" (The value must be one of Ulayer.PLOT_ENLARGE_*)
- "VARIABLE_TEXT_HEIGHT"
- "VARIABLE_TEXT_DIRECTION" (The value must be one of Ulayer.PLOT_VARTEXT_DIRECTION_*)

sDefValue The default value

Returns:

value of the plot parameter or default value

void grabWidget ( )

Prints out the name of the widget that is clicked with the mouse.

void gridAlign (double *dStep*)

Align to grid

Parameters:

dStep grid size

void gridCross (boolean *bCross*)

Sets the grid crosses or lines displayed when the grid is visible

Parameters:

bCross `true` show the grid crosses; `false` show the grid lines

boolean gridCross ( )

Returns `true` if the grid crosses are displayed or not

Returns:

`true` if the grid crosses are displayed or `false` if the grid lines are displayed

void gridOrigin (double *ptOrigin_x*,
double *ptOrigin_y*)

Sets the grid origin to given **Point**

Parameters:

ptOrigin_x (X coordinate) new grid origin **Point**
ptOrigin_y (Y coordinate) new grid origin **Point**

void gridOrigin (Point *ptOrigin*)

Sets the grid origin to given **Point**

Parameters:

ptOrigin new grid origin **Point**

Point gridOrigin ( )

Returns the current grid origin **Point**

Returns:

the current grid origin **Point**

void gridOutline (double *refPoint_x*,
double *refPoint_y*,
double *offset_x*,
double *offset_y*,
int *repeatX*,
int *repeatY*)

PCB images in a flat data need to be outlined. We can manually construct outline contour in this (usually the layer in plane 1 and outline extra layer) layer according to a PCB image in reference layer. We mark reference point in PCB image and give an offset of the same reference data in direction X and Y. The repeat parameter

defines a number of outlines in each direction.

Parameters:

refPoint_x	(X coordinate) A point where the PCB data are taken as an reference (for alignment)
refPoint_y	(Y coordinate) A point where the PCB data are taken as an reference (for alignment)
offset_x	(X coordinate) of the outline grid in X and Y
offset_y	(Y coordinate) of the outline grid in X and Y
repeatX	a number of the outlines in X direction
repeatY	a number of the outlines in Y direction

Returns:

Uapeobj can be `null` in case the new outline couldn't be created. The return aperture can be the same in case the target is without rotation. Only the object(flash) is the new. In case there is a rotation the return object is completely new aperture with the (one) new flash.

**void gridOutline (Point *refPoint,*
Point *offset,*
int *repeatX,*
int *repeatY*
)**

PCB images in a flat data need to be outlined. We can manually construct outline contour in this (usually the layer in plane 1 and outline extra layer) layer according to a PCB image in reference layer. We mark reference point in PCB image and give an offset of the same reference data in direction X and Y. The repeat parameter defines a number of outlines in each direction.

Parameters:

refPoint	A point where the PCB data are taken as an reference (for alignment)
offset	of the outline grid in X and Y
repeatX	a number of the outlines in X direction
repeatY	a number of the outlines in Y direction

Returns:

Uapeobj can be `null` in case the new outline couldn't be created. The return aperture can be the same in case the target is without rotation. Only the object(flash) is the new. In case there is a rotation the return object is completely new aperture with the (one) new flash.

**void gridStep (double *dStepX,*
double *dStepY*
)**

Sets the X and Y distances between grid crosses or lines

Parameters:

dStepX	the X distance between grid crosses or lines
dStepY	the Y distance between grid crosses or lines

Point gridStep ( )

Returns the **Point** with coordinates presenting the X and Y distance between grid crosses or lines

Returns:

the **Point** with coordinates presenting the X and Y distance between grid crosses or lines

double gridStepX ( )

Returns the X distance between grid crosses or lines

Returns:
the X distance between grid crosses or lines

double gridStepY ( )

Returns the Y distance between grid crosses or lines

Returns:
the Y distance between grid crosses or lines

void gridVisible (boolean *bVisible*)

Sets the grid visible or hidden

Parameters:
bVisible `true` to make the grid visible, `false` to make it hidden

boolean gridVisible ( )

returns `true` if the grid is currently visible

Returns:
`true` if the grid is currently visible

void groupApeBy (String *spec*)

Group apertures with common specs on all active layers

Parameters:
spec Spec: "number", "definition" or "plane"

void groupApertureDefinitions ( )

Aperture Manager: Group Apertures by Definition in active layer in plane 1

void groupApertureNumbers ( )

Aperture Manager: Group Apertures by Numbers in active layer in plane 1

void groupAperturesByPolarity ( )

Aperture Manager: Group Apertures by Polarity in active layer in plane 1

double groupUFD (double *dDistance*)

Group faults in a fault list

Parameters:

dDistance - neighborhood

Returns:

true value of grouping distance

void groupUFD ( )

Group faults in a fault list

void helpOnContext ( )

Get help on context.

void hideAll ( )

Hide all layers

void hideBlockStructure ( )

hide Block Structure Information dialog

void HitachiSpotDiameterCompensation (double *dOffset,*
double *dArcExpandMarginOverrideMicrons,*
int *iMode,*
int *iFastMode,*
boolean *bAmSkipFlag,*
boolean *bChangePolarity*
)

Performs Hitachi SPP

Parameters:

dOffset	compensation value in microns (positive = normal (thin); negative = reverse (thicken))
dArcExpandMarginOverrideMicrons	if >= 0, override ucam.db arc expand margin (in micron)

iMode	SPP mode (can be 1 or 2)
iFastMode	SPP fast mode flag (can be 0 or 1 or 2)
bAmSkipFlag	whether or not to skip processing of Aperture Macros (ignored)
bChangePolarity	nonzero indicates that the layer will be reversed by the DE system; therefore, the sign of the offset should swapped when working in Mode 2

void HitachiSpotDiameterCompensation (double *dOffset,*
double *dArcExpandMarginOverrideMicrons,*
int *iMode,*
boolean *bFastMode,*
boolean *bAmSkipFlag,*
boolean *bChangePolarity*
)

Performs Hitachi SPP

Parameters:

dOffset	compensation value in microns (positive = normal (thin); negative = reverse (thicken))
dArcExpandMarginOverrideMicrons	if >= 0, override ucam.db arc expand margin (in microns)
iMode	SPP mode (can be 1 or 2)
bFastMode	SPP fast mode flag
bAmSkipFlag	whether or not to skip processing of Aperture Macros (ignored)
bChangePolarity	nonzero indicates that the layer will be reversed by the DE system; therefore, the sign of the offset should swapped when working in Mode 2

String i8Jobarticleid ( )

Returns the current job articleid.

Returns:
current job articleid or null if there is no current job

String i8JobBoardid ( )

Returns the current job boardid.

Returns:
current job boardid or null if there is no current job

String i8JobCustomer ( )

Returns the current job customer.

Returns:
current job customer or null if there is no current job

boolean i8Jobdelete ( )

Deletes the current job.

Returns:
true if the job was successfully deleted, false otherwise

int i8JobDuration ( )

Returns the current job duration.

Returns:
current job duration or -1 if there is no current job

Date i8JobFinishtime ( )

Returns the current job finishtime.

Returns:
current job finishtime or null if there is no current job

int i8JobFullduration ( )

Returns the current job fullduration.

Returns:
current job fullduration or -1 if there is no current job

int i8JobId ( )

Returns the current job id.

Returns:
current job id or -1 if there is no current job

String i8JobLocation ( )

Returns the current job location.

Returns:
current job location or null if there is no current job

int i8JobPriority ( )

Returns the current job priority.

Returns:

current job priority or -1 if there is no current job

String i8JobProgress ( )

Returns the current job progress.

Returns:
current job progress or null if there is no current job

int i8JobQueueposition ( )

Returns the current job queueposition.

Returns:
current job queueposition or -1 if there is no current job

Date i8JobStarttime ( )

Returns the current job starttime.

Returns:
current job starttime or null if there is no current job

Date i8JobSubmittime ( )

Returns the current job submittime.

Returns:
current job submittime or null if there is no current job

void importEpc (String *sPath*)

import Epc CAR job

Parameters:
sPath - full path to 'job'.car file inside EPC files directory

int importExternal (String *sExtName*,
String *sWheName*,
String *sLanguage*,
boolean *bKeepExtension*,
String *sLayClass*,
String *sLayAtt*,
String *sStatus*,
String *sWheLang*,
boolean *bAnalyzed*,

```
                String    sWheFile,
                int       iLocale
              )
```

import External file

Parameters:

sExtName	external file full path
sWheName	wheel file full path
sLanguage	language
bKeepExtension	keep extension
sLayClass	layer class
sLayAtt	layer attributes
sStatus	status
sWheLang	wheel language
bAnalyzed	analyzed flag
sWheFile	wheel file needed info ("---" or "...")
iLocale	file location 1=local 0=remote

Returns:
status

**int importExternal (String *sExtName,*
String *sWheName,*
String *sLanguage,*
boolean *bKeepExtension*
)**

import External file

Parameters:

sExtName	external file full path
sWheName	wheel file full path
sLanguage	language
bKeepExtension	keep extension

Returns:
status

int importExternal (String *sExtName*)

import External file, it will call format analyzer

Parameters:

sExtName external file full path

Returns:
status

void importFile (String *sScriptPath*)

Import script from file.

Parameters:
sScriptPath - full path to the script file

void importGwk (String *sPath*)

import GWK file

Parameters:
sPath - full path to GWK file

String importHeptaCSV (String *sCSVFile*,
String *sDXFFile*,
String *sOptions*
)

Parameters:
sCSVFile csv file
sDXFFile dxf file
sOptions options for hepta csv

Returns:
`null` if OK otherwise return s error message as a string

void importHouei (String *sPath*)

import Houei job

Parameters:
sPath - full path to Houei kend_xxx.par file inside Houei job directory

void importIpc (String *sPath*,
String *sVersion*
)

import MET and IPC356 and IPC356b file

Parameters:
sPath - full path to MET or IPC file
sVersion - version of IPC format (correct values are "ipc356" or "ipc356b" or "met");

void importIPC2581 (String *sPath*,
String *sStep*,
String *sLayer*
)

import IPC2581

Parameters:

sPath path to IPC2581 file
sStep step name
sLayer layer name

void importODBxx (String *sPath,*
String *sStep,*
String *sLayer,*
ObjectList *oReplaceCodeMap*
)

import ODBxx

Parameters:

sPath path to ODB job
sStep step name
sLayer layer name
oReplaceCodeMap the replace code map ObjectList

Example: [{ [{"$$CODE","Result text1"}], [{"$$CODE1","Result text2"}] }]

void importODBxx (String *sPath,*
String *sStep,*
String *sLayer*
)

import ODBxx

Parameters:

sPath path to ODB job
sStep step name
sLayer layer name

void importODBxx (String *sPath,*
String *sStep,*
ObjectList *oReplaceCodeMap*
)

import ODBxx

Parameters:

sPath path to ODB job
sStep ODB step
oReplaceCodeMap the replace code map ObjectList

Example: [{ [{"$$CODE","Result text1"}], [{"$$CODE1","Result text2"}] }]

void importODBxx (String *sPath,*
String *sStep*

)

import ODBxx

Parameters:

sPath path to ODB job
sStep ODB step

void importPolarBuildup (String *sPolarFilePath*)

Imports the buildup information from a stkx file generated by the Polar application.

Parameters:

sPolarFilePath The file path specification for the Polar stkx file.

void importScript (String *sScript*)

Import given script

Parameters:

sScript - the script text

void importWf (String *sPath*)

import WF file

Parameters:

sPath - full path to WF file

void innerCopperCount (boolean *bUseMask*,
boolean *bConfirmMaskUsage*
)

Calculates the copper surface in active inner layers.

Parameters:

bUseMask When true, active mask layers are taken into account: The "free of mask" area's are then calculated. The mask with attachment top is used for the top outer layer. The mask with attachment bottom is used for the bottom outer layer. The mask with attachment none is used for inner layers.

bConfirmMaskUsage When true and active mask layers exist, then asks for confirmation if active mask layers should be used, or not.

void innerCopperCount ( )

Calculates the copper surface in active inner layers.

```
void insertAperture ( boolean    bBefore,
                      String     sSrcLayer,
                      ObjectList srcApeIndex
                    )
```

Aperture Manager: Insert aperture(s) of (other) layer near current aperture

Parameters:

bBefore	true: insert before apeIndex; false: insert after apeIndex
sSrcLayer	Name of the layer to take apertures from
srcApeIndex	Array of indexes of the apertures on the source layer

```
int insertArc ( double arc_fx,
                double arc_fy,
                double arc_tx,
                double arc_ty,
                double arc_cx,
                double arc_cy,
                String arc_sense,
                int    iNet,
                String sSelection
              )
```

Insert arc using current aperture

Parameters:

arc_fx	(from X coordinate) The arc
arc_fy	(from Y coordinate) The arc
arc_tx	(to X coordinate) The arc
arc_ty	(to Y coordinate) The arc
arc_cx	(to X coordinate) The arc
arc_cy	(to Y coordinate) The arc
arc_sense	(arc sense) The arc
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

Returns:

Error status. (0 on success, non-zero on failure.)

```
int insertArc ( Arc    arc,
                int    iNet,
                String sSelection
              )
```

Insert arc using current aperture

Parameters:

arc	The arc
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as

selected.

Returns:
Error status. (0 on success, non-zero on failure.)

```
void insertArc3Point ( double  pnt1_x,
                       double  pnt1_y,
                       double  pnt2_x,
                       double  pnt2_y,
                       double  pnt3_x,
                       double  pnt3_y
                     )
```

Insert arc using current aperture. Count arc from 3 outline points.

Parameters:
pnt1_x (X coordinate) outline point (from point)
pnt1_y (Y coordinate) outline point (from point)
pnt2_x (X coordinate) outline point
pnt2_y (Y coordinate) outline point
pnt3_x (X coordinate) outline point (to point)
pnt3_y (Y coordinate) outline point (to point)

```
void insertArc3Point ( Point  pnt1,
                       Point  pnt2,
                       Point  pnt3
                     )
```

Insert arc using current aperture. Count arc from 3 outline points.

Parameters:
pnt1 outline point (from point)
pnt2 outline point
pnt3 outline point (to point)

```
void insertArc3Point ( double  pnt1_x,
                       double  pnt1_y,
                       double  pnt2_x,
                       double  pnt2_y,
                       double  pnt3_x,
                       double  pnt3_y,
                       int     iNet,
                       String  sSelection
                     )
```

Insert arc using current aperture. Count arc from 3 outline points.

Parameters:
pnt1_x (X coordinate) outline point (from point)

pnt1_y	(Y coordinate) outline point (from point)
pnt2_x	(X coordinate) outline point
pnt2_y	(Y coordinate) outline point
pnt3_x	(X coordinate) outline point (to point)
pnt3_y	(Y coordinate) outline point (to point)
iNet	The net number of the object.
sSelection	The selection option. If "sel" is specified, the object is marked as selected.

```
void insertArc3Point ( Point  pnt1,
                       Point  pnt2,
                       Point  pnt3,
                       int    iNet,
                       String sSelection
                     )
```

Insert arc using current aperture. Count arc from 3 outline points.

Parameters:

pnt1	outline point (from point)
pnt2	outline point
pnt3	outline point (to point)
iNet	The net number of the object.
sSelection	The selection option. If "sel" is specified, the object is marked as selected.

```
void insertArcCenterStart ( double pntCenter_x,
                            double pntCenter_y,
                            double pntFrom_x,
                            double pntFrom_y,
                            double pntTo_x,
                            double pntTo_y,
                            String sDirection
                          )
```

Insert arc using current aperture. Count arc from 2 outline points and center point.

Parameters:

pntCenter_x	(X coordinate) center point
pntCenter_y	(Y coordinate) center point
pntFrom_x	(X coordinate) outline point
pntFrom_y	(Y coordinate) outline point
pntTo_x	(X coordinate) outline point
pntTo_y	(Y coordinate) outline point
sDirection	direction of arc

```
void insertArcCenterStart ( Point  pntCenter,
                            Point  pntFrom,
                            Point  pntTo,
                            String sDirection
```

)

Insert arc using current aperture. Count arc from 2 outline points and center point.

Parameters:

pntCenter	center point
pntFrom	outline point
pntTo	outline point
sDirection	direction of arc

void insertArcCenterStart (**double** ***pntCenter_x,***
double ***pntCenter_y,***
double ***pntFrom_x,***
double ***pntFrom_y,***
double ***pntTo_x,***
double ***pntTo_y,***
String ***sDirection,***
int ***iNet,***
String ***sSelection***
)

Insert arc using current aperture. Count arc from 2 outline points and center point.

Parameters:

pntCenter_x	(X coordinate) center point
pntCenter_y	(Y coordinate) center point
pntFrom_x	(X coordinate) outline point
pntFrom_y	(Y coordinate) outline point
pntTo_x	(X coordinate) outline point
pntTo_y	(Y coordinate) outline point
sDirection	direction of arc
iNet	The net number of the object.
sSelection	The selection option. If "sel" is specified, the object is marked as selected.

void insertArcCenterStart (**Point** ***pntCenter,***
Point ***pntFrom,***
Point ***pntTo,***
String ***sDirection,***
int ***iNet,***
String ***sSelection***
)

Insert arc using current aperture. Count arc from 2 outline points and center point.

Parameters:

pntCenter	center point
pntFrom	outline point
pntTo	outline point
sDirection	direction of arc
iNet	The net number of the object.
sSelection	The selection option. If "sel" is specified, the object is marked as selected.

boolean insertArcConcentric (boolean *bSelection*,
ObjectList *oArcs*
)

Insert concentric arcs using current aperture

Parameters:

bSelection The selection. If "true" is specified, the objects are marked as selected.
oArcs The array of arcs

Returns:

Error status. (true, something was added)

void insertBreak (Point *line_fp*,
Point *line_tp*
)

Add break to arcs and draws on the line.

Parameters:

line_fp (from point) The break will be on the line
line_tp (to point) The break will be on the line

void insertBreak (double *line_fx*,
double *line_fy*,
double *line_tx*,
double *line_ty*
)

Add break to arcs and draws on the line.

Parameters:

line_fx (from X coordinate) The break will be on the line
line_fy (from Y coordinate) The break will be on the line
line_tx (to X coordinate) The break will be on the line
line_ty (to Y coordinate) The break will be on the line

void insertBreak (Line *line*)

Add break to arcs and draws on the line.

Parameters:

line The break will be on the line

void insertContourText (double *rect_xmin*,
double *rect_ymin*,

```
                    double  rect_xmax,
                    double  rect_ymax,
                    String  sText,
                    String  sFontName,
                    int     iFontStyle,
                    String  sMirror,
                    boolean bReverse,
                    boolean bAllowDistortion,
                    String  sSelection
                )
```

Creates and add to the layer a contour aperture containing the specified text.

Parameters:

rect_xmin	(left boundary of rectangle) The enclosing rectangle for the text.
rect_ymin	(bottom boundary of rectangle) The enclosing rectangle for the text.
rect_xmax	(right boundary of rectangle) The enclosing rectangle for the text.
rect_ymax	(top boundary of rectangle) The enclosing rectangle for the text.
sText	The text to be imaged.
sFontName	font name, see java.awt.Font constructor
iFontStyle	font style, see java.awt.Font constructor
sMirror	The mirror setting, either "", "X", "Y" or "XY".
bReverse	Indication whether the contour aperture needs to be reversed or not.
bAllowDistortion	Allowed distortion if set to true
sSelection	if set to "sel", the contour text will be selected

```
void insertContourText ( Rectangle rect,
                         String    sText,
                         String    sFontName,
                         int       iFontStyle,
                         String    sMirror,
                         boolean   bReverse,
                         boolean   bAllowDistortion,
                         String    sSelection
                       )
```

Creates and add to the layer a contour aperture containing the specified text.

Parameters:

rect	The enclosing rectangle for the text.
sText	The text to be imaged.
sFontName	font name, see java.awt.Font constructor
iFontStyle	font style, see java.awt.Font constructor
sMirror	The mirror setting, either "", "X", "Y" or "XY".
bReverse	Indication whether the contour aperture needs to be reversed or not.
bAllowDistortion	Allowed distortion if set to true
sSelection	if set to "sel", the contour text will be selected

```
void insertCopper ( int    number,
                    String attach,
```

```
                String material,
                double thickness,
                String reference,
                double tolerance,
                String supplier
              )
```

Inserts copper with a given material specification to the specified Layer.

Parameters:

number	Signal Layer index
attach	attach "top" or "bottom"
material	Material name
thickness	Material thickness
reference	Material reference
tolerance	Material tolerance
supplier	Material supplier

```
void insertCore ( int        layNum,
                  boolean    matTop,
                  boolean    matBot,
                  String     material,
                  String     topMaterial,
                  String     botMaterial,
                  double     thickness,
                  double     topThickness,
                  double     botThickness,
                  String     reference,
                  double     tolerance,
                  double     erConstant,
                  String     supplier,
                  ObjectList attrNames,
                  Object[]   attrValues,
                  boolean    revInsert
                )
```

Inserts core with a given material specification between the specified Layers.

Parameters:

layNum	int Top layer index
matTop	boolean true if the core is attached to the top layer
matBot	boolean true if the core is attached to the bottom layer
material	String Material name
topMaterial	String material on the top of the core
botMaterial	String material on the bot of the core
thickness	double Material thickness
topThickness	double thickness of the top material
botThickness	double thickness of the bot material
reference	String Material reference
tolerance	double Material tolerance
erConstant	double Material ER constant
supplier	String Material supplier

attrNames	String list of the Material attribute names
attrValues	String the same sized list of the Material attribute values. If the Material attribute has no value put the empty string "", no `null` value!
revInsert	boolean toggles top and bottom material.

void insertCore (int ***layNum,***
boolean ***matTop,***
boolean ***matBot,***
String ***material,***
String ***topMaterial,***
String ***botMaterial,***
double ***thickness,***
double ***topThickness,***
double ***botThickness,***
String ***reference,***
double ***tolerance,***
double ***erConstant,***
String ***supplier,***
ObjectList ***attrNames,***
Object[] ***attrValues***
)

Inserts core with a given material specification between the specified Layers.

Parameters:

layNum	int Top layer index
matTop	boolean true if the core is attached to the top layer
matBot	boolean true if the core is attached to the bottom layer
material	String Material name
topMaterial	String material on the top of the core
botMaterial	String material on the bot of the core
thickness	double Material thickness
topThickness	double thickness of the top material
botThickness	double thickness of the bot material
reference	String Material reference
tolerance	double Material tolerance
erConstant	double Material ER constant
supplier	String Material supplier
attrNames	String list of the Material attribute names
attrValues	String the same sized list of the Material attribute values. If the Material attribute has no value put the empty string "", no `null` value!

void insertCore (int ***layTop,***
int ***layBot,***
boolean ***matTop,***
boolean ***matBot,***
String ***material,***
String ***topMaterial,***
String ***botMaterial,***

```
                double      thickness,
                double      topThickness,
                double      botThickness,
                String      reference,
                double      tolerance,
                double      erConstant,
                String      supplier,
                ObjectList  attrNames,
                Object[]    attrValues,
                boolean     revInsert
              )
```

Inserts core with a given material specification between the specified Layers.

Parameters:

layTop	int Top layer index
layBot	int Bottom layer index
matTop	boolean true if the core is attached to the top layer
matBot	boolean true if the core is attached to the bottom layer
material	String Material name
topMaterial	String material on the top of the core
botMaterial	String material on the bot of the core
thickness	double Material thickness
topThickness	double thickness of the top material
botThickness	double thickness of the bot material
reference	String Material reference
tolerance	double Material tolerance
erConstant	double Material ER constant
supplier	String Material supplier
attrNames	String list of the Material attribute names
attrValues	String the same sized list of the Material attribute values. If the Material attribute has no value put the empty string "", no `null` value!
revInsert	boolean toggles top and bottom material.

```
void insertCore ( int         layTop,
                  int         layBot,
                  boolean     matTop,
                  boolean     matBot,
                  String      material,
                  String      topMaterial,
                  String      botMaterial,
                  double      thickness,
                  double      topThickness,
                  double      botThickness,
                  String      reference,
                  double      tolerance,
                  double      erConstant,
                  String      supplier,
                  ObjectList  attrNames,
                  Object[]    attrValues
                )
```

Inserts core with a given material specification between the specified Layers.

Parameters:

layTop	int Top layer index
layBot	int Bottom layer index
matTop	boolean true if the core is attached to the top layer
matBot	boolean true if the core is attached to the bottom layer
material	String Material name
topMaterial	
botMaterial	
thickness	double Material thickness
topThickness	
botThickness	
reference	String Material reference
tolerance	double Material tolerance
erConstant	double Material ER constant
supplier	String Material supplier
attrNames	String list of the Material attribute names
attrValues	String the same sized list of the Material attribute values. If the Material attribute has no value put the empty string "", no `null` value!

```
int insertDraw ( double  line_fx,
                 double  line_fy,
                 double  line_tx,
                 double  line_ty,
                 int     iNet,
                 String  sSelection
               )
```

Insert draw using current aperture

Parameters:

line_fx	(from X coordinate) The line
line_fy	(from Y coordinate) The line
line_tx	(to X coordinate) The line
line_ty	(to Y coordinate) The line
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

Returns:

Error status. (0 on success, non-zero on failure.)

```
int insertDraw ( Line    line,
                 int     iNet,
                 String  sSelection
               )
```

Insert draw using current aperture

Parameters:

line	The line

iNet The net number of the object.
sSelection The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

Returns:
Error status. (0 on success, non-zero on failure.)

int insertDraw (double *line_fx*,
double *line_fy*,
double *line_tx*,
double *line_ty*
)

Insert draw using current aperture

Parameters:
line_fx (from X coordinate) The line
line_fy (from Y coordinate) The line
line_tx (to X coordinate) The line
line_ty (to Y coordinate) The line

Returns:
Error status. (0 on success, non-zero on failure.)

int insertDraw (Line *line*)

Insert draw using current aperture

Parameters:
line The line

Returns:
Error status. (0 on success, non-zero on failure.)

int insertFlash (double *pt_x*,
double *pt_y*,
int *iNet*,
String *sSelection*
)

Insert Flash using current aperture, set net number and selection

Parameters:
pt_x (X coordinate) The flash point
pt_y (Y coordinate) The flash point
iNet The net number of the object.
sSelection The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

Returns:
Error status. (0 on success, non-zero on failure.)

```
int insertFlash ( Point  pt,
                  int    iNet,
                  String sSelection
                )
```

Insert Flash using current aperture, set net number and selection

Parameters:

pt	The flash point
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

Returns:

Error status. (0 on success, non-zero on failure.)

```
int insertFlash ( double pt_x,
                  double pt_y
                )
```

Insert Flash using current aperture

Parameters:

pt_x	(X coordinate) The flash point
pt_y	(Y coordinate) The flash point

Returns:

Error status. (0 on success, non-zero on failure.)

```
int insertFlash ( Point pt )
```

Insert Flash using current aperture

Parameters:

pt	The flash point

Returns:

Error status. (0 on success, non-zero on failure.)

```
void insertFlashRepeat ( double pt_x,
                         double pt_y,
                         int    iNx,
                         double dXstep,
                         int    iNy,
                         double dYstep,
                         String sSelection
                       )
```

Generates a step and repeat of flashes

Parameters:

pt_x	(X coordinate) The offset (start) point.
pt_y	(Y coordinate) The offset (start) point.
iNx	The number of repetitions in x direction.
dXstep	The step in x direction.
iNy	The number of repetitions in y direction.
dYstep	The step in y direction.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the flashes are marked as selected.

```
void insertFlashRepeat ( Point  pt,
                         int    iNx,
                         double dXstep,
                         int    iNy,
                         double dYstep,
                         String sSelection
                       )
```

Generates a step and repeat of flashes

Parameters:

pt	The offset (start) point.
iNx	The number of repetitions in x direction.
dXstep	The step in x direction.
iNy	The number of repetitions in y direction.
dYstep	The step in y direction.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the flashes are marked as selected.

```
void insertFlashRepeat ( double pt_x,
                         double pt_y,
                         int    iNx,
                         double dXstep,
                         int    iNy,
                         double dYstep
                       )
```

Generates a step and repeat of flashes

Parameters:

pt_x	(X coordinate) The offset (start) point.
pt_y	(Y coordinate) The offset (start) point.
iNx	The number of repetitions in x direction.
dXstep	The step in x direction.
iNy	The number of repetitions in y direction.
dYstep	The step in y direction.

```
void insertFlashRepeat ( Point  pt,
                         int    iNx,
```

```
                    double  dXstep,
                    int     iNy,
                    double  dYstep
                )
```

Generates a step and repeat of flashes

Parameters:

pt	The offset (start) point.
iNx	The number of repetitions in x direction.
dXstep	The step in x direction.
iNy	The number of repetitions in y direction.
dYstep	The step in y direction.

```
void insertFullArc3Point ( double  pnt1_x,
                           double  pnt1_y,
                           double  pnt2_x,
                           double  pnt2_y,
                           double  pnt3_x,
                           double  pnt3_y
                         )
```

Insert full arc using current aperture. Count arc from 3 outline points.

Parameters:

pnt1_x	(X coordinate) outline point
pnt1_y	(Y coordinate) outline point
pnt2_x	(X coordinate) outline point
pnt2_y	(Y coordinate) outline point
pnt3_x	(X coordinate) outline point
pnt3_y	(Y coordinate) outline point

```
void insertFullArc3Point ( Point  pnt1,
                           Point  pnt2,
                           Point  pnt3
                         )
```

Insert full arc using current aperture. Count arc from 3 outline points.

Parameters:

pnt1	outline point
pnt2	outline point
pnt3	outline point

```
void insertFullArc3Point ( double  pnt1_x,
                           double  pnt1_y,
                           double  pnt2_x,
                           double  pnt2_y,
                           double  pnt3_x,
```

```
                        double  pnt3_y,
                        int     iNet,
                        String  sSelection
                      )
```

Insert full arc using current aperture. Count arc from 3 outline points.

Parameters:

pnt1_x	(X coordinate) outline point
pnt1_y	(Y coordinate) outline point
pnt2_x	(X coordinate) outline point
pnt2_y	(Y coordinate) outline point
pnt3_x	(X coordinate) outline point
pnt3_y	(Y coordinate) outline point
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

```
void insertFullArc3Point ( Point   pnt1,
                           Point   pnt2,
                           Point   pnt3,
                           int     iNet,
                           String  sSelection
                         )
```

Insert full arc using current aperture. Count arc from 3 outline points.

Parameters:

pnt1	outline point
pnt2	outline point
pnt3	outline point
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

```
void insertFullArcCenterRadius ( double  pntCenter_x,
                                 double  pntCenter_y,
                                 double  dRadius,
                                 String  sDirection
                               )
```

Insert full arc using current aperture. Count arc from radius and center point.

Parameters:

pntCenter_x	(X coordinate) center point
pntCenter_y	(Y coordinate) center point
dRadius	radius of arc
sDirection	direction of arc

```
void insertFullArcCenterRadius ( Point   pntCenter,
```

```
                                double  dRadius,
                                String  sDirection
                              )
```

Insert full arc using current aperture. Count arc from radius and center point.

Parameters:

pntCenter	center point
dRadius	radius of arc
sDirection	direction of arc

```
void insertFullArcCenterRadius ( double  pntCenter_x,
                                 double  pntCenter_y,
                                 double  dRadius,
                                 String  sDirection,
                                 int     iNet,
                                 String  sSelection
                               )
```

Insert full arc using current aperture. Count arc from radius and center point.

Parameters:

pntCenter_x	(X coordinate) center point
pntCenter_y	(Y coordinate) center point
dRadius	radius of arc
sDirection	direction of arc
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

```
void insertFullArcCenterRadius ( Point   pntCenter,
                                 double  dRadius,
                                 String  sDirection,
                                 int     iNet,
                                 String  sSelection
                               )
```

Insert full arc using current aperture. Count arc from radius and center point.

Parameters:

pntCenter	center point
dRadius	radius of arc
sDirection	direction of arc
iNet	The net number of the object.
sSelection	The selection option. Either "all" or "sel". If "sel" is specified, the object is marked as selected.

```
boolean insertParallel ( boolean     bSelection,
                         ObjectList  oLines
```

)

Insert parallel draws using current aperture

Parameters:

bSelection The selection. If "true" is specified, the objects are marked as selected.
oLines The array of draws

Returns:

Error status. (true, something was added)

void insertPolydrawRect (double *rect_xmin,*
double *rect_ymin,*
double *rect_xmax,*
double *rect_ymax,*
boolean *bSel*
)

Insert polydraw rectangle using current aperture

Parameters:

rect_xmin (left boundary of rectangle) rectangle to draw
rect_ymin (bottom boundary of rectangle) rectangle to draw
rect_xmax (right boundary of rectangle) rectangle to draw
rect_ymax (top boundary of rectangle) rectangle to draw
bSel if `true` selects the new created objects

void insertPolydrawRect (Rectangle *rect,*
boolean *bSel*
)

Insert polydraw rectangle using current aperture

Parameters:

rect rectangle to draw
bSel if `true` selects the new created objects

void insertPolydrawRect (double *rect_xmin,*
double *rect_ymin,*
double *rect_xmax,*
double *rect_ymax*
)

Insert polydraw rectangle using current aperture

Parameters:

rect_xmin (left boundary of rectangle) rectangle to draw
rect_ymin (bottom boundary of rectangle) rectangle to draw
rect_xmax (right boundary of rectangle) rectangle to draw
rect_ymax (top boundary of rectangle) rectangle to draw

void insertPolydrawRect (Rectangle *rect*)

Insert polydraw rectangle using current aperture

Parameters:

rect rectangle to draw

void insertPolydrawRect (double *drawRectangle_xmin*,
double *drawRectangle_ymin*,
double *drawRectangle_xmax*,
double *drawRectangle_ymax*,
boolean *bRectCW*,
boolean *bSel*
)

Insert polydraw rectangle using current aperture

Parameters:

drawRectangle_xmin	(left boundary of rectangle) rectangle to draw
drawRectangle_ymin	(bottom boundary of rectangle) rectangle to draw
drawRectangle_xmax	(right boundary of rectangle) rectangle to draw
drawRectangle_ymax	(top boundary of rectangle) rectangle to draw
bRectCW	set true if the rectangle should be CW
bSel	set true if the rectangle should be selected

void insertPolydrawRect (Rectangle *drawRectangle*,
boolean *bRectCW*,
boolean *bSel*
)

Insert polydraw rectangle using current aperture

Parameters:

drawRectangle	rectangle to draw
bRectCW	set true if the rectangle should be CW
bSel	set true if the rectangle should be selected

void insertPolydrawRect (Point *p1*,
Point *p2*,
boolean *bRectCW*,
boolean *bSel*
)

Deprecated:

Insert polydraw rectangle using current aperture

Parameters:

p1	bottom left point of the rectangle

p2	top right point of the rectangle
bRectCW	set true if the rectangle should be CW
bSel	set true if the rectagle should be selected

```
void insertPolygon ( boolean    bSelection,
                     ObjectList polygon
                   )
```

Insert polygon using current aperture

Parameters:

bSelection	The selection. If "true" is specified, the objects are marked as selected.
polygon	polygon as array of lines

```
void insertPrePreg ( int        topLayer,
                     int        bottomLayer,
                     String     sPosition,
                     String     material,
                     double     thickness,
                     String     reference,
                     double     tolerance,
                     double     erConstant,
                     String     supplier,
                     ObjectList attrNames,
                     Object[]   attrValues
                   )
```

Inserts prepreg with a given specification between the specified Layers.

Parameters:

topLayer	top Layer for prepreg
bottomLayer	bottom Layer for prepreg
sPosition	"Up" or "Down" value
material	String Material name
thickness	double Material thickness
reference	String Material reference
tolerance	double Material tolerance
erConstant	double Material ER constant
supplier	String Material supplier
attrNames	String list of the Material attribute names
attrValues	String the same sized list of the Material attribute values. If the Material attribute has no value put the empty string "", no `null` value!

```
void insertTab ( double p_x,
                 double p_y,
                 double dis,
                 String pat
               )
```

Replaces a part of a track or a part of a corner formed by two tracks by a predefined pattern. This pattern is actually a predefined DPF-job. You must position the pattern in the required position using the Numbers dialog box or by inserting it as a flash. The use on corners is limited to corners of 90 degrees, formed by a horizontal and a vertical tracks. The patterns are flashed on the corner point (for corners) or on the click point for single tracks.

Parameters:

p_x	(X coordinate) tab location
p_y	(Y coordinate) tab location
dis	the distance of the opening in the rout where the tab will be placed
pat	the current pattern

```
void insertTab ( Point  p,
                 double dis,
                 String pat
               )
```

Replaces a part of a track or a part of a corner formed by two tracks by a predefined pattern. This pattern is actually a predefined DPF-job. You must position the pattern in the required position using the Numbers dialog box or by inserting it as a flash. The use on corners is limited to corners of 90 degrees, formed by a horizontal and a vertical tracks. The patterns are flashed on the corner point (for corners) or on the click point for single tracks.

Parameters:

p	tab location
dis	the distance of the opening in the rout where the tab will be placed
pat	the current pattern

```
void insertVectorText ( double pt_x,
                        double pt_y,
                        String sText,
                        String sFont,
                        double dWidth,
                        double dSpacing,
                        String sMirror,
                        double dRotation,
                        double dScale
                      )
```

Insert Vector Text using current aperture

Parameters:

pt_x	(X coordinate) flash point
pt_y	(Y coordinate) flash point
sText	Text string
sFont	Vector Text font
dWidth	Character width (for fixed width fonts like "ODBstandard")
dSpacing	Character spacing (for variable width fonts like "vtx")
sMirror	Mirror (none, X, Y or XY)
dRotation	Rotation
dScale	Scale in both directions

```
void insertVectorText ( Point  pt,
                        String sText,
                        String sFont,
                        double dWidth,
                        double dSpacing,
                        String sMirror,
                        double dRotation,
                        double dScale
                      )
```

Insert Vector Text using current aperture

Parameters:

pt	flash point
sText	Text string
sFont	Vector Text font
dWidth	Character width (for fixed width fonts like "ODBstandard")
dSpacing	Character spacing (for variable width fonts like "vtx")
sMirror	Mirror (none, X, Y or XY)
dRotation	Rotation
dScale	Scale in both directions

```
void insertVectorText ( double pt_x,
                        double pt_y,
                        String sText,
                        String sFont,
                        double dWidth,
                        double dSpacing,
                        String sMirror,
                        double dRotation,
                        double dScaleX,
                        double dScaleY
                      )
```

Insert Vector Text using current aperture

Parameters:

pt_x	(X coordinate) flash point
pt_y	(Y coordinate) flash point
sText	Text string
sFont	Vector Text font
dWidth	Character width (for fixed width fonts like "ODBstandard")
dSpacing	Character spacing (for variable width fonts like "vtx")
sMirror	Mirror (none, X, Y or XY)
dRotation	Rotation
dScaleX	Scale in X direction
dScaleY	Scale in Y direction

```
void insertVectorText ( Point  pt,
```

```
                    String  sText,
                    String  sFont,
                    double  dWidth,
                    double  dSpacing,
                    String  sMirror,
                    double  dRotation,
                    double  dScaleX,
                    double  dScaleY
                )
```

Insert Vector Text using current aperture

Parameters:

pt	flash point
sText	Text string
sFont	Vector Text font
dWidth	Character width (for fixed width fonts like "ODBstandard")
dSpacing	Character spacing (for variable width fonts like "vtx")
sMirror	Mirror (none, X, Y or XY)
dRotation	Rotation
dScaleX	Scale in X direction
dScaleY	Scale in Y direction

```
void intersectDraws ( double  pt_x,
                      double  pt_y
                    )
```

Add intersection point on 2 intersecting draws

Parameters:

pt_x	(X coordinate) Intersection location
pt_y	(Y coordinate) Intersection location

```
void intersectDraws ( Point  pt  )
```

Add intersection point on 2 intersecting draws

Parameters:

pt	Intersection location

```
boolean isDirectory ( ObjectList  fileInfo  )
```

Tests whether the file denoted by this fileInfo is a directory.

Parameters:

fileInfo	objectlist with the file information

Returns:

true if and only if the file denoted by this fileInfo is a directory; false otherwise

See also:

HSH_base::osFileInfo(String)

boolean isEqual (Object *oParam1,*
Object *oParam2*
)

Compare 2 given objects

Parameters:

oParam1 it can be any of supported object
oParam2 it can be any of supported object

Returns:

true if both are the same

boolean isFile (ObjectList *fileInfo*)

Tests whether the file denoted by this abstract is a normal file. A file is normal if it is not a directory and, in addition, satisfies other system-dependent criteria.

Parameters:

fileInfo objectlist with the file information

Returns:

`true` if the file is normal file; `false` otherwise

See also:

HSH_base::osFileInfo(String)

boolean isHidden (ObjectList *fileInfo*)

Tests whether the file denoted by this fileInfo is a hidden file. The exact definition of hidden is system-dependent. On UNIX systems, a file is considered to be hidden if its name begins with a period character ('.'). On Microsoft Windows systems, a file is considered to be hidden if it has been marked as such in the filesystem.

Parameters:

fileInfo objectlist with the file information

Returns:

`true` if and only if the file denoted by this fileInfo is hidden according to the conventions of the underlying platform

See also:

HSH_base::osFileInfo(String)

boolean isLayerInPlane (int *iPlaneNumber*)

Returns `true` when the layer in given plane exists, otherwise returns `false`

Parameters:

iPlaneNumber plane number

Returns:
`true` when the layer in given plane exists, otherwise returns `false`

void job_save_shm_and_release (String *sShmName*)

save current job to shared memory

Parameters:
sShmName

int jobApeMaxNumber ( )

Gets the highest aperture number.

Returns:
The highest aperture number used so far. If no aperture is found, -1 is returned.

String jobATEMachine ( )

Returns the file specification of the ATE tester file.

Returns:
the file specification of the ATE tester file.

**void jobAttribute (String *name,*
String *value*
)**

Sets the given value to the given job attribute. If the attribute exists, its value is changed to the new value. Otherwise the attribute is created. If the value is `null` the attribute with the given name is removed.

Parameters:
name the job attribute name
value the job attribute value

String jobAttribute (String *name*)

Returns the value of the Job attribute with given name.

Parameters:
name the job attribute name

Returns:
the value of the Job attribute with given name or null if the attribute is not defined in the job.

String jobAttribute ( )

Returns comma separated list of the all Job attributes. **Example:** "uJobSize=mil,5840,2048,**adjacency.distance**=0,adjacency.blindnetsubst=1,uSessionPaiFile=D:.pai "

Returns:

comma separated list of the all Job attributes.

**void jobCopperCount (boolean *bUseMask,*
boolean *bConfirmMaskUsage*
)**

Calculates the copper surface in job (all layers).

Parameters:

bUseMask	When true, active mask layers are taken into account: The "free of mask" area's are then calculated. The mask with attachment top is used for the top outer layer. The mask with attachment bottom is used for the bottom outer layer. The mask with attachment none is used for inner layers.
bConfirmMaskUsage	When true and active mask layers exist, then asks for confirmation if active mask layers should be used, or not.

void jobCopperCount ( )

Calculates the copper surface in job (all layers).

void jobCustomer (String *sCustomer*)

Sets the customer this job belongs to.

Parameters:

sCustomer the customer this job belongs to

String jobCustomer ( )

Returns the customer this job belongs to.

Returns:

the customer this job belongs to.

void jobDRCParameters (String *sDrc*)

Sets the design rule check parameter in this job.

Parameters:

sDrc the design rule check parameter file entry in this job.

String jobDRCParameters ( )

Returns the design rule check parameter file entry in this job.

Returns:
the design rule check parameter file entry in this job.

Rectangle jobEnclosingBox ( )

Gets the enclosing rectangle of the job.

Returns:
Job enclosing rectangle.

void jobExtension (String *sExtension*)

Sets the extension of the job.

Parameters:
sExtension

void jobFixture (String *sFixture*)

Sets the fixture type used in electrical testing functions.

Parameters:
sFixture the fixture type. Either "top", "bot", "ssa" or "dsa".

String jobFixture ( )

Returns the fixture type used in the electrical test functions. Possible values are "top", "bot", "ssa" and "dsa".

Returns:
the fixture type used in the electrical test functions. Possible values are "top", "bot", "ssa" and "dsa".

boolean jobHasPattern (boolean *bUsed*)

Returns `true` if the job has an aperture with a pattern in active layers.

Parameters:
bUsed pattern is used when it affects the image.

Returns:
`true` if the job has an aperture with a pattern in active layers.

void jobInfo (String[] *sInfo*)

Sets the information associated with the job.

Parameters:

sInfo the new information associated with the job.

void jobInfo (String *sInfo*)

Sets the information associated with the job.

Parameters:

sInfo the new information associated with the job.

String jobInfo ( )

Returns the information associated with this object.

Returns:

the information associated with this object.

void jobLayMask (String *sLayMask*)

Sets the layer mask parameter for the Job. If the layer mask is already set in the Job, the function takes all active layer having current layer mask and replaces the layer name with the new layer mask.

Example:

```
String cadDir = "HOME:\\Data\\Cad";


openJob(cadDir + "\\cad.job");


String layMask = jobLayMask();
String layMask = jobLayMask();


activateAllLayers();


jobLayMask("cad_");
jobLayMask("cad_");

layMask = jobLayMask();
layMask = jobLayMask();



jobLayMask("Lepa_");
jobLayMask("Lepa_");

String trgDir = cadDir + "\\copy";
osMkDir(trgDir);

saveJobAs(trgDir + "\\lepa.job");
```

Parameters:

sLayMask the layer mask that will be set in the Job.

String jobLayMask ( )

Gets the layer mask currently set in the Job parameters.

Returns:
the layer mask currently set in the Job parameters.

See also:
jobLayMask(String)

jobLayMask(String)

int jobMaxNetnumer ( )

Gets the maximum netnumber in the job.

Returns:
the maximum netnumber in the job.

void jobName (String *sName*)

Sets the job name.

Parameters:
sName new name.

String jobName ( )

Returns the job name.

Returns:
the job name.

boolean jobNetlist ( )

Checks if all positive data has a netnumber. It skips the layers of class JL_EXTRA. It is not required to set the layer(s) active.

Returns:
true if all positive data has a netnumber.

int jobNumApes ( )

Returns the number of the apertures in the job.

Returns:
the number of the apertures in the job.

int jobNumBothExtras ( )

Returns the number of the extra layers in the job with the both attach.

Returns:
the number of the extra layers in the job with the both attach.

int jobNumBothExtras (String *subClass*)

Returns the number of the extra layers in the job with the both attach.

Parameters:
subClass The subclass for the layers wanted. If not important specify "". The default offered subclasses are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

Returns:
the number of the extra layers in the job with the both attach.

int jobNumBottomExtras ( )

Returns the number of the extra layers in the job with the bottom attach.

Returns:
the number of the extra layers in the job with the bottom attach.

int jobNumBottomExtras (String *subClass*)

Returns the number of the extra layers in the job with the bottom attach.

Parameters:
subClass The subclass for the layers wanted. If not important specify "". The default offered subclasses are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

Returns:
the number of the extra layers in the job with the bottom attach.

int jobNumCores ( )

Returns the number of the cores in the job.

Returns:
the number of the cores in the job.

int jobNumDrills ( )

Returns the number of the drill layers in the job.

Returns:
the number of the drill layers in the job.

int jobNumDrills (String *subClass*)

Returns the number of the drill layers in the job.

Parameters:

subClass The subclass for the layer wanted. If not important specify "". The default offered subclasses are "drill", "buried", "blind", "plated", "unplated" and "fixing".

Returns:

the number of the drill layers in the job.

int jobNumExtras ( )

Returns the number of the extra layers in the job.

Returns:

the number of the extra layers in the job.

int jobNumExtras (String *subClass*)

Returns the number of the extra layers in the job.

Parameters:

subClass The subclass for the layers wanted. If not important specify "". The default offered subclasses are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

Returns:

the number of the extra layers in the job.

int jobNumLayers ( )

Returns the number of the all layers in the job.

Returns:

the number of the all layers in the job.

int jobNumNoneExtras ( )

Returns the number of the extra layers in the job with the none attach.

Returns:

the number of the extra layers in the job with the none attach.

int jobNumNoneExtras (String *subClass*)

Returns the number of the extra layers in the job with the none attach.

Parameters:

subClass The subclass for the layers wanted. If not important specify "". The default offered

subclasses are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

Returns:
the number of the extra layers in the job with the none attach.

int jobNumPrepregs (int *start*)

Returns the number of the prepregs in the job between given layer and the following layer.

Parameters:
start the layer index where prepreg counting starts.

Returns:
the number of the prepregs in the job between given layer and the following layer.

int jobNumPrepregs ( )

Returns the number of the prepregs in the job.

Returns:
the number of the prepregs in the job.

int jobNumSignals ( )

Returns the number of the signal layers in the job.

Returns:
the number of the signal layers in the job.

int jobNumSignals (String *subClass*)

Returns the number of the signal layers in the job.

Parameters:
subClass The subclass for the layer wanted. If not important specify "". The default offered subclasses are "outer", "inner" and "mixed".

Returns:
the number of the signal layers in the job.

int jobNumTopExtras ( )

Returns the number of the extra layers in the job with the top attach.

Returns:
the number of the extra layers in the job with the top attach.

int jobNumTopExtras (String *subClass*)

Returns the number of the extra layers in the job with the top attach.

Parameters:

subClass The subclass for the layers wanted. If not important specify "". The default offered subclasses are "rout", "silk", "mask", "paste", "exclusion", "netref", "testpoints", "probe" and "guideplate".

Returns:

the number of the extra layers in the job with the top attach.

void jobPath (String *sPath*)

Sets the path of the job.

Parameters:

sPath the path of the job

String jobPath ( )

Returns the path of the job.

Returns:

Gets the path of the job.

void jobRevision (String *sRevision*)

Sets the job revision.

Parameters:

sRevision the job revision

String jobRevision ( )

Returns the the job revision.

Returns:

Gets the job revision.

int jobSelectCount (String *sOption*)

Returns the number of selected objects in a job.

Parameters:

sOption Specifies the type of the objects to count. "a" for arcs, "f" for flashes, "d" for draws, "r" for regions and "v" for vector text.

Returns:

the number of selected objects.

int jobSelectCount ( )

Returns the number of selected objects in a job.

Returns:
the number of selected objects.

boolean jobSelection ( )

Checks if this job contains selected items.

Returns:
true if there are selections, false otherwise.

Rectangle jobSelectionEnclosingBox ( )

Gets the enclosing rectangle of the job selection.

Returns:
Job selection enclosing rectangle.

void jobSize (double *pntSize_x*,
double *pntSize_y*)

Sets job size.

Parameters:
pntSize_x (X coordinate) new size of job
pntSize_y (Y coordinate) new size of job

void jobSize (Point *pntSize*)

Sets job size.

Parameters:
pntSize new size of job

void jobSize (String *sUnit*,
double *pntSize_x*,
double *pntSize_y*)

Sets job size.

Parameters:

sUnit unit of size
pntSize_x (X coordinate) new size of job
pntSize_y (Y coordinate) new size of job

void jobSize (String *sUnit,*
Point *pntSize*
)

Sets job size.

Parameters:
sUnit unit of size
pntSize new size of job

void jobSize (String *sSize*)

Sets job size.

Parameters:
sSize string like "mil,0,0", there are unit, x size and y size

Point jobSize ( )

Returns job size using uJobSize attribute value.

Returns:
Point representing job size, x and y size.

void jobSpec (String *sSpec*)

Sets the full file specification of a job.

Parameters:
sSpec the full file specification of a job.

String jobSpec ( )

Returns the full file specification of a job.

Returns:
the full file specification of a job.

void jobUserData (String *sUserData*)

Sets the user data of a job.

Parameters:

sUserData the user data of a job.

String jobUserData ( )

Returns the user data of a job.

Returns:
the user data of a job.

void lajCleanLegendLayer (boolean *DoMask*,
double *MaskClearance*,
boolean *DoCu*,
double *CuClearance*,
boolean *DoCuPads*,
double *CuPadClearance*,
boolean *bDoCuFOM*,
double *iCuFOMClearance*,
boolean *bDoCuPadsFOM*,
double *iCuPadsFOMClearance*,
boolean *doPlatedDrills*,
double *platedDrillClearance*,
boolean *doUnplatedDrills*,
double *UnplatedDrillClearance*,
boolean *DoSmallDraws*,
double *MinDrawSize*
)

Cleans all active silk layers against all active matching copper + drill + mask layers with the presented clearances

Parameters:

DoMask	Clip around mask
MaskClearance	Min. mask clearance
DoCu	Clip around outer layer
CuClearance	Min. outer clearance
DoCuPads	Clip around copper pads
CuPadClearance	Min. copper pad clearance
bDoCuFOM	Clip around copper not covered by mask
iCuFOMClearance	Min. unmasked copper clearance
bDoCuPadsFOM	Clip around copper pads not covered by mask
iCuPadsFOMClearance	Min. unmasked copper pad clearance
doPlatedDrills	Clip around plated drills
platedDrillClearance	Min. plated drill clearance
doUnplatedDrills	Clip around unplated drills
UnplatedDrillClearance	Min. unplated drill clearance
DoSmallDraws	Remove small objects
MinDrawSize	Min. object size

void lajDefineWord ( )

Turns selected objects on all active layers into a word

void lajDeselectAllWords ( )

Deselect all textboxes on all active layers

void lajDragWord (double *pt_x*,
double *pt_y*,
double *radius*,
double *offset_x*,
double *offset_y*,
double *limit*,
boolean *enforcelimit*
)

Drag move a word, used in mousetool

Parameters:

pt_x	(X coordinate) **Point** where word is
pt_y	(Y coordinate) **Point** where word is
radius	Radius in which to search for a word
offset_x	(X coordinate) Contains horizontal and vertical distance to move
offset_y	(Y coordinate) Contains horizontal and vertical distance to move
limit	maximal distance word can be moved
enforcelimit	whether to enforce the distance limit

void lajDragWord (Point *pt*,
double *radius*,
Point *offset*,
double *limit*,
boolean *enforcelimit*
)

Drag move a word, used in mousetool

Parameters:

pt	**Point** where word is
radius	Radius in which to search for a word
offset	Contains horizontal and vertical distance to move
limit	maximal distance word can be moved
enforcelimit	whether to enforce the distance limit

void lajLegendDRC (boolean *bDoLineWidth*,
double *dMinLineWidth*,
boolean *bDoMask*,
double *dMaskClearance*,

```
                    boolean  bDoCu,
                    double   dCuClearance,
                    boolean  bDoCuPads,
                    double   dCuPadClearance,
                    boolean  bDoCuFOM,
                    double   dCuFOMClearance,
                    boolean  bDoCuPadsFOM,
                    double   dCuPadsFOMClearance,
                    boolean  bDoPlatedDrills,
                    double   dPlatedDrillClearance,
                    boolean  bDoUnplatedDrills,
                    double   dUnplatedDrillClearance,
                    boolean  bDoSmallDraws,
                    double   dMinDrawSize
                  )
```

Does DRC check on legend layers against other active layers

Parameters:

bDoLineWidth	Check line width
dMinLineWidth	Min. line width
bDoMask	Check mask clearance
dMaskClearance	Min. mask clearance
bDoCu	Check outer clearance
dCuClearance	Min. outer clearance
bDoCuPads	Check copper pad clearance
dCuPadClearance	Min. copper pad clearance
bDoCuFOM	Check clearance to copper free of mask
dCuFOMClearance	Min. unmasked copper clearance
bDoCuPadsFOM	Check clearance to copper pads free of mask
dCuPadsFOMClearance	Min. unmasked copper pad clearance
bDoPlatedDrills	Check clearance to plated drills
dPlatedDrillClearance	Min. plated drill clearance
bDoUnplatedDrills	Check clearance to unplated drills
dUnplatedDrillClearance	Min. unplated drill clearance
bDoSmallDraws	Check for small objects
dMinDrawSize	Min. object size

```
void lajLegendTextToWords ( double  dMaxSize,
                            int     iMaxSpacing
                          )
```

Converts text to rectangles, splitting vertical and horizontal text, on all active silk layers

Parameters:

dMaxSize	maximal text size
iMaxSpacing	maximal letter spacing, in percent of average letter width

```
void lajMoveWord ( String  value,
                   double  dx,
```

```
                    double dy,
                    double limit
                )
```

Move word with given uText value over distance dx, dy, unless the distance is larger than limit and enforcelimit is true

Parameters:

value uText attribute value of the word to be moved
dx horizontal distance
dy vertical distance
limit maximal distance word can be moved

```
void lajMoveWord ( String value,
                    double dx,
                    double dy
                )
```

Move word with given uText value over distance dx, dy

Parameters:

value uText attribute value of the word to be moved
dx horizontal distance
dy vertical distance

```
void lajScaleWord ( String value,
                    double factor,
                    double limit
                )
```

Scale the word with the give uText value with the given factor, but don't make apertures smaller than limit

Parameters:

value uText attribute value of the word to be scaled
factor scale factor
limit min. aperture size after scaling

```
void lajScaleWord ( String value,
                    double factor
                )
```

Scale the word with the give uText value with the given factor

Parameters:

value uText attribute value of the word to be scaled
factor scale factor

```
void lajScaleWordOnPt ( double pt_x,
```

```
                        double   pt_y,
                        double   radius,
                        double   scale,
                        double   limit,
                        boolean  enforcelimit
                      )
```

Scale word nearest to point with given factor, keeping aperture larger than min. size. Used in mousetool.

Parameters:

pt_x	(X coordinate) **Point** where word is
pt_y	(Y coordinate) **Point** where word is
radius	Radius in which to search for a word
scale	factor by which to scale word
limit	min. aperture width after scaling
enforcelimit	whether to enforce the aperture size limit

```
void lajScaleWordOnPt ( Point    pt,
                        double   radius,
                        double   scale,
                        double   limit,
                        boolean  enforcelimit
                      )
```

Scale word nearest to point with given factor, keeping aperture larger than min. size. Used in mousetool.

Parameters:

pt	**Point** where word is
radius	Radius in which to search for a word
scale	factor by which to scale word
limit	min. aperture width after scaling
enforcelimit	whether to enforce the aperture size limit

```
void lajSelectAllWords (  )
```

Select all textboxes on all active layers

```
void lajUndefineWord (  )
```

Removes uText attribute from selected objects on active layers

```
void layActive ( ObjectList  layerID,
                 boolean     bActive
               )
```

Sets activity on the Layer defined by layer ID

Parameters:

layerID the layer ID e.g. from **getLayers()** function

bActive `true` if the layer defined by layer ID will be set as active; `false` for remove activity on a layer defined by the layer ID

See also:

getLayers()

boolean layActive (ObjectList *layerID*)

Returns activity of the Layer with the given layer ID

Parameters:

layerID the layer ID e.g. from **getLayers()** function

Returns:

`true`/`false` activity on a layer defined by the layer ID

See also:

getLayers()

void layActive (boolean *bActive*)

Sets activity on the current Layer

Parameters:

bActive `true` if the current layer will be set as active; `false` for remove activity on the current layer

boolean layActive ( )

Returns activity of the current Layer

Returns:

`true`/`false` activity of the current Layer

void layAlias (String *sAlias*)

Sets new alias to the current Layer.

Parameters:

sAlias String new alias of the current Layer.

String layAlias ( )

Returns current Layer alias.

Returns:

current Layer alias

int layApeCount ( )

Return numbers of aperture on current layer

Returns:
Number of apertures on current layer. If the layer does not exist return -1

void layAttach (String *sAttach*)

Sets the attachment of the extra layer.

Parameters:
sAttach possible values are "top", "bottom", "none" or "both".

String layAttach ( )

Gets the attachment of the extra layer.

Returns:
"top", "bottom", "none" or "both".

**void layAttribute (String *name,*
String *value*
)**

Sets the given value to the given layer attribute. If the attribute exists, its value is changed to the new value. Otherwise the attribute is created. If the value is `null` the attribute with the given name is removed.

Parameters:
name the layer attribute name
value the layer attribute value

String layAttribute (String *name*)

Returns the value of the layer attribute with given name.

Parameters:
name the layer attribute name

Returns:
the value of the layer attribute with given name or null if the attribute is not defined in the layer.

String layAttribute ( )

Returns comma separated list of the all layer attributes. **Example:** "uMatReference=R-5715-4,uMatTolerance=0.0076,uMatSupplier=Ucamco,form="

Returns:
comma separated list of the all layer attributes.

void layClass (String *sNewClass*)

Changes class of the current layer to the new class.

Parameters:

sNewClass Layer class, possible values are "layer", "drill" or "extra".

String layClass ( )

Gets the class of current layer as a String.

Returns:

"layer", "drill" or "extra".

**void layCopperCount (boolean *bUseMask,*
boolean *bConfirmMaskUsage*
)**

Calculates the copper surface in active layers.

Parameters:

bUseMask	When true, active mask layers are taken into account: The "free of mask" area's are then calculated. The mask with attachment top is used for the top outer layer. The mask with attachment bottom is used for the bottom outer layer. The mask with attachment none is used for inner layers.
bConfirmMaskUsage	When true and active mask layers exist, then asks for confirmation if active mask layers should be used, or not.

void layCopperCount ( )

Calculates the copper surface in active layers.

Rectangle layEnclosingBox ( )

Gets the enclosing rectangle of the current layer.

Returns:

Layer enclosing rectangle.

void layerViewSplit (boolean *on*)

Set variable to enable/diable displaying split lines in the Job View dialog Call repaint for the dialog Job View

Parameters:

on if true will be Job View will be set to Split View

**int layExtractPlotStamps (String *dstLayName,*
String *sOptions,*
ObjectList *sFilters*
)**

extract plotstamps into another layer, keep aperture an object attributes

Parameters:

dstLayName destination layer name (if null or "" or layer with given name does not exist, returns only count)

sOptions "TEXT" or "VTXT" or "BLO" or combination like "TEXT,VTX,BLO", also flag "DELETE" to delete original objects and "KEEPLINK" to keep link for crosslinked blocks

sFilters list of strings to match when extracting (if no match, no extraction)(null=empty=no Filter=extract all) **Example:** [{"%Level", "%BatchNo"}]

Returns:

nr. of found objects

void layFrom (int *layFrom*)

Sets the layer number where the drill holes start.

Parameters:

layFrom layer number where the drill holes start.

int layFrom ( )

Gets the layer number where the drill holes start.

Returns:

the layer number where the drill holes start.

boolean layHasPattern (boolean *bUsed*)

Returns `true` if the layer has an aperture with a pattern.

Parameters:

bUsed pattern is used when it affects the image.

Returns:

`true` if the layer has an aperture with a pattern.

ObjectList layID ( )

Returns ObjectArray with the current layer specification

Returns:

ObjectArray with layer specification (name, class, subclass, attachment, index, activity, number of apertures) **Example:** [{"cad_m1", "extra", "mask", "bottom", 1, true, -1}]

See also:
getLayers()

void layIndex (int *iIndex*)

Sets the index of the current layer in the array of layers in the job this layer belongs to.

Parameters:
iIndex the new index of the current layer in the array of layers in the job this layer belongs to.

int layIndex ( )

Gets the index of the current layer in the array of layers in the job this layer belongs to.

Returns:
the index of the current layer in the array of layers in the job this layer belongs to. -1 if the current layer is not set.

void layInfo (String *sText*)

Sets the information to the current layer

Parameters:
sText String containing the information about the layer

String layInfo ( )

Gets the information associated with the current layer.

Returns:
the information associated with the current layer. `null` if the current layer is not set.

void layMaterial (String *sMaterial*)

Sets the layer material (extra or layer class).

Parameters:
sMaterial the new layer material.

String layMaterial ( )

Gets the layer material (extra or layer class).

Returns:
the layer material (extra or layer class). null if the current Layer is not set.

void layName (String *sName*)

Sets new name to the current Layer.

Parameters:

sName String new name of the current Layer.

String layName ( )

Returns current Layer name.

Returns:

current Layer name

void layNumber (int *iNumber*)

Sets the layer number in the group of extra layers with the same class, subclass and attach.

Parameters:

iNumber the new layer number in the group of extra layers with the same class, subclass and attach.

int layNumber ( )

Gets the layer number in the group of extra Layers with the same class, subclass and attach.

Returns:

the layer number in the group of extra Layers with the same class, subclass and attach. -1 if the current Layer is not set.

void layReadable (String *sSide*)

Sets the side where this layer is readable.

Parameters:

sSide String "top" or "bottom".

String layReadable ( )

Gets the side where this layer is readable.

Returns:

"top" or "bottom".

void layReverse (boolean *bReverse*)

Sets the reverse flag for the current Layer.

Parameters:

bReverse boolean true if the layer has negative data. (Its polarity is opposite compared to the produced copper image.)

boolean layReverse ( )

Has the layer negative data?

Returns:

true if the layer has negative data. (Its polarity is opposite compared to the produced copper image.)

boolean laySelection ( )

Checks if the current Layer contains selected items.

Returns:

`true` if there are selections, `false` otherwise.

Rectangle laySelectionEnclosingBox ( )

Gets the enclosing rectangle of the selection of the current layer.

Returns:

Layer selection enclosing rectangle.

void laySubClass (String *sSubClass*)

Sets the subclass of current layer.

Parameters:

sSubClass String the default subclasses offered by Ucam are "rout", "silk", "mask", "paste", "exclusion", "netref", "frame", "testpoints", "guideplate" and "probe".

String laySubClass ( )

Gets the subclass of current layer. The default subclasses offered by Ucam are "rout", "silk", "mask", "paste", "exclusion", "netref", "frame", "testpoints", "guideplate" and "probe".

Returns:

the subclass of current layer eg. "rout", "silk", "mask", "paste", "exclusion", "netref", "frame", "testpoints", "guideplate" and "probe".

void layThickness (double *dThickness*)

Sets the thickness of the current layer.

Parameters:

dThickness double the thickness of the current layer.

double layThickness ( )

Gets the thickness of the current layer.

Returns:
the thickness of the current layer.

void layTo (int *layTo*)

Sets the layer number where the drill holes end.

Parameters:
layTo layer number where the drill holes end.

int layTo ( )

Gets the layer number where the drill holes end.

Returns:
the layer number where the drill holes end.

double layZPos ( )

Return the x position of the current layer.

Returns:
the x position of the current layer.

void liftUpUpcbBlocks ( )

Lift up blocks with uPcb attribute to the begin of the layer. Image is kept.

Line Line (double *ptFromX*,
double *ptFromY*,
double *ptToX*,
double *ptToY*,
String *units*
)

Create line from four coordinates

Parameters:
ptFromX start point x coordinate
ptFromY start point y coordinate
ptToX line end point x coordinate

ptToY line end point y coordinate
units Ucam units

Returns:
the line

Line Line (double *ptFromX*,
double *ptFromY*,
double *ptToX*,
double *ptToY*
)

Create line from four coordinates

Parameters:
ptFromX start point x coordinate
ptFromY start point y coordinate
ptToX line end point x coordinate
ptToY line end point y coordinate

Returns:
the line

Line Line (Line *line*)

Create copy of a line

Parameters:
line original line

Returns:
the line

Line Line (Point *ptFrom*,
Point *ptTo*
)

Create line from two points

Parameters:
ptFrom line start point
ptTo line end point

Returns:
the line

ObjectList listFrames ( )

Lists names of all custom dialogs

Returns:

Object Array with custom dialogs names. Name can be used in openFrame command.

See also:

openFrame(String)

void loadApertures (String *sDpfFile*)

Aperture Manager: Load Apertures from a DPF File

Parameters:

sDpfFile DPF File to read Apertures from

void loadBuildup (String *buildupSpec*)

Loads the buildup from the given .jot file into the current job. The layer names of the current job are overwritten.

Parameters:

buildupSpec the .jot file path with the buildup content to be loaded.

**void loadFrames (boolean *bVerbose,*
boolean *bLoadOnce*
)**

Loads all custom dialogs (dialogs extending CustomFrame class).

Parameters:

bVerbose detailed information about loading is visible if set to `true`

bLoadOnce Load dialog only once if set to `true`. A `false` value is useful when a dialog is under development and needs to be loaded every time the command is used in script.

See also:

CustomFrame

void loadFrames ( )

Loads all custom dialogs (dialogs extending CustomFrame class).

See also:

loadFrames(boolean, boolean)

loadFrames(boolean, boolean)

void loadSplitConfig (String *sConfigName*)

Load split configuration for the given name

Parameters:

sConfigName name of configuration

void loadUFD (String *sUFDName*)

Loads all the faults from the given file into the current fault database.

Parameters:

sUFDName UFD file specification

void loadWorkspace (String *sWorkspaceName*)

Discard modifications and reload current layout. NOTE: The same as menu command Workspaces > *workspace_name*

Parameters:

sWorkspaceName

void loadWorkspace ( )

Discard modifications and reload current layout. NOTE: The same as menu command Workspaces > Reload

double maxInvalidArcsDeviation ( )

Find and return maximal deviation of all (selected) invalid arcs The deviation is from distance Center point to To point or Center point to From point Current units will be used

Returns:

maximal deviation in current units of all (selected) invalid arcs

int measureFingers (String *szOption*)

Measure gold fingers (ASE)

Parameters:

szOption The given options: "attribute" or "selection"

Returns:

0: ok, 1: an error occurred

void measureLayers ( )

Measure dimensions of active selection

void measureObjects (double *p1_x,*
double *p1_y,*
double *p2_x,*

```
                    double p2_y
                )
```

Measure clearance between objects Results are displayed in Numbers dialog

Parameters:

p1_x (X coordinate) flashpoint of first object
p1_y (Y coordinate) flashpoint of first object
p2_x (X coordinate) flashpoint of second object
p2_y (Y coordinate) flashpoint of second object

```
void measureObjects ( Point p1,
                      Point p2
                    )
```

Measure clearance between objects Results are displayed in Numbers dialog

Parameters:

p1 flashpoint of first object
p2 flashpoint of second object

```
void measurePoints ( double pt_x,
                     double pt_y
                   )
```

Sets the point1 and point2 in Numbers dialog as the same point

Parameters:

pt_x (X coordinate) first and the same second point
pt_y (Y coordinate) first and the same second point

```
void measurePoints ( Point pt )
```

Sets the point1 and point2 in Numbers dialog as the same point

Parameters:

pt first and the same second point

```
void measurePoints ( double p1_x,
                     double p1_y,
                     double p2_x,
                     double p2_y
                   )
```

Measure clearance between points Results are displayed in Numbers dialog

Parameters:

p1_x (X coordinate) first object
p1_y (Y coordinate) first object

p2_x (X coordinate) second object
p2_y (Y coordinate) second object

void measurePoints (Point *p1,*
Point *p2*
)

Measure clearance between points Results are displayed in Numbers dialog

Parameters:
p1 first object
p2 second object

void mergeContours ( )

Merge Contours

void mergeContoursSingle ( )

Merge Contours Single

void mergeContoursSingleAdd ( )

Merge Contours Single Add

void mergeLayers (String *posNegAlt,*
boolean *delLay*
)

Merge Layer(s)

Parameters:
posNegAlt Merge option ("Positive", "Negative" or "Alternate")
delLay if true, delete merged layers

void mirror (String *axis,*
boolean *bUseCenter,*
boolean *bOnRefPoints*
)

Mirror selections around axis

Parameters:
axis Value of the axis (X or Y)
bUseCenter If true, mirror is done around center

bOnRefPoints If true, mirror is also applied to reference points

void models (String *sModelShape,*
double *dTolerance*
)

Models replace selected painted shape on active layers by standard or complex shape

Parameters:

sModelShape Shape of selected model: standard or complex
dTolerance tolerance to be used when searching for model instances

See also:

models(String)

models(String)

void models (String *sModelShape*)

Models replace selected painted shape on active layers by standard or complex shape try count the best tolerance to be used when searching for model instances.

Parameters:

sModelShape Shape of selected model: standard or complex

See also:

models(String, double)

models(String, double)

boolean modelsCreateComplex ( )

modelsCreateComplex Create complex aperture with the same shape like selected objects. the function **modelsDefineSelections()** must be call before

Returns:

a status, if the complex aperture was created, the status is true

See also:

models(String)

models(String, double)

boolean modelsCreateStandard (double *dTolerance*)

modelsCreateStandard Try find a standard aperture looks like selected object. the function **modelsDefineSelections()** must be call before

Parameters:

dTolerance a tolerance

Returns:

a status, if a aperture was found, the status is true

See also:

models(String)

models(String, double)

Rectangle modelsDefineSelections ( )

modelsDefineSelections Define selected part of active layer in plane 1. This function must be run before other functions from the dialog models.

Returns:

enclosing box (rectangle) of the temporary layer

See also:

models(String)

models(String, double)

int modelsReplace (double *pntTolerance_x*, double *pntTolerance_y*)

modelsReplace Replace selected objects by created models in active layer in plane 1. the function `modelsDefineSelections()` must be call before

Parameters:

pntTolerance_x (X coordinate) a tolerance

pntTolerance_y (Y coordinate) a tolerance

Returns:

returns a count of replaced apertures; -1 an error

See also:

models(String)

models(String, double)

int modelsReplace (Point *pntTolerance*)

modelsReplace Replace selected objects by created models in active layer in plane 1. the function `modelsDefineSelections()` must be call before

Parameters:

pntTolerance a tolerance

Returns:

returns a count of replaced apertures; -1 an error

See also:

models(String)

models(String, double)

int modelsSelect (double *pntTolerance_x*,
double *pntTolerance_y*
)

modelsSelect Select object. the function **modelsDefineSelections()** must be call before

Parameters:

pntTolerance_x (X coordinate) a tolerance
pntTolerance_y (Y coordinate) a tolerance

Returns:

number of selected objects

See also:

models(String)

models(String, double)

int modelsSelect (Point *pntTolerance*)

modelsSelect Select object. the function **modelsDefineSelections()** must be call before

Parameters:

pntTolerance a tolerance

Returns:

number of selected objects

See also:

models(String)

models(String, double)

void modifyCore (int *iTopLay*,
String *sAtt*,
int *iNewTopLay*,
int *iNewBotLay*,
String *sNewAtt*,
double *dThickness*,
String *sMaterial*,
String *sInfo*
)

Modify Core

Parameters:

iTopLay	Top Layer index the Core is connected to
sAtt	The attachment for this core.
iNewTopLay	The new top Layer index the Core is connected to
iNewBotLay	The new bottom Layer index the Core is connected to
sNewAtt	Defines the attachment for this core "top", "bottom", "both" or "none".
dThickness	Defines the thickness for this core.
sMaterial	Defines the material for this core.

sInfo Defines the description/info for this core.

```
void modifyDrill ( String  sName,
                   String  sAlias,
                   String  sClass,
                   String  sSubClass,
                   int     iFrom,
                   int     iTo,
                   double  dThickness
                 )
```

Modify Drill

Parameters:

sName new layer name
sAlias new alias
sClass class of the changed layer - "Layer", "Drill" or "Extra"
sSubClass subclass of the changed layer

- **for class "Layer"** - "outer", "inner", "mixed"
- **for class "Drill"** - "drill", "buried", "blind", "plated", "unplated", "fixing"
- **for class "Extra"** - "rout", "score", "outline", "mask", "silk", "clipping", "coupon", "netref", "testpoints", "midpoints", "probe", "guideplate", "frame", "bolref", "bolinsp", "resistor", "dro", "snapshot", "help", "adjacency"

iFrom index of the first drilled layer from top
iTo index of the last drilled layer from top
dThickness thickness of the modified layer

```
void modifyExtra ( String  sName,
                   String  sAlias,
                   String  sClass,
                   String  sSubClass,
                   String  sAttach,
                   int     iNumber,
                   boolean bReverse,
                   String  sMaterial
                 )
```

Modify Extra layer

Parameters:

sName new layer name
sAlias new alias
sClass class of the changed layer - "Layer", "Drill" or "Extra"
sSubClass subclass of the changed layer

- **for class "Layer"** - "outer", "inner", "mixed"
- **for class "Drill"** - "drill", "buried", "blind", "plated", "unplated", "fixing"
- **for class "Extra"** - "rout", "score", "outline", "mask", "silk", "clipping", "coupon", "netref", "testpoints", "midpoints", "probe", "guideplate", "frame", "bolref", "bolinsp", "resistor", "dro", "snapshot", "help", "adjacency"

sAttach	Attach - "top" or "bottom", "none"
iNumber	new position of the modified layer
bReverse	true - modified layer will be marked as reverse
sMaterial	Name of the material

void modifyFeedback (String *sName,*
String *sAlias,*
String *sClass,*
String *sSubClass,*
String *sAttach*
)

Modify Feedback layer

Parameters:

sName	new layer name
sAlias	new alias
sClass	class of the changed layer - "Layer", "Drill" or "Extra"
sSubClass	subclass of the changed layer

- **for class "Layer"** - "outer", "inner", "mixed"
- **for class "Drill"** - "drill", "buried", "blind", "plated", "unplated", "fixing"
- **for class "Extra"** - "rout", "score", "outline", "mask", "silk", "clipping", "coupon", "netref", "testpoints", "midpoints", "probe", "guideplate", "frame", "bolref", "bolinsp", "resistor", "dro", "snapshot", "help", "adjacency"

sAttach	Attach - "top" or "bottom", "none"

void modifyLayer (String *sName,*
String *sAlias,*
String *sClass,*
String *sSubClass,*
int *iNumber,*
boolean *bReverse,*
double *dZPosition,*
String *sReadable,*
String *sMaterial,*
double *dThickness*
)

Modify signal Layer

Parameters:

sName	new layer name
sAlias	new alias
sClass	class of the changed layer - "Layer", "Drill" or "Extra"
sSubClass	subclass of the changed layer

- **for class "Layer"** - "outer", "inner", "mixed"
- **for class "Drill"** - "drill", "buried", "blind", "plated", "unplated", "fixing"
- **for class "Extra"** - "rout", "score", "outline", "mask", "silk", "clipping", "coupon", "netref", "testpoints", "midpoints", "probe", "guideplate", "frame", "bolref", "bolinsp",

"resistor", "dro", "snapshot", "help", "adjacency"

iNumber	new position of the modified layer
bReverse	true - modified layer will be marked as reverse
dZPosition	ZPozition
sReadable	Readable side "top" or "bottom"
sMaterial	Name of the material
dThickness	thickness of the modified layer

void modifyPrePreg (int *iTopLay*,
int *iIndex*,
int *iNewTopLay*,
int *iNewBotLay*,
int *iNewIndex*,
double *dThickness*,
String *sMaterial*,
String *sInfo*
)

Modify PrePreg

Parameters:

iTopLay	Top Layer index the PrePreg is connected to
iIndex	Index of the PrePreg in PrePreg stack up.
iNewTopLay	The new top Layer index the PrePreg is connected to
iNewBotLay	The new bottom Layer index the PrePreg is connected to
iNewIndex	The new index of the PrePreg in PrePreg stack up.
dThickness	Defines the thickness for this PrePreg.
sMaterial	Defines the material for this PrePreg.
sInfo	Defines the description/info for this PrePreg.

void move (double *pt_x*,
double *pt_y*,
boolean *bOnRefPoints*
)

Move (selected) object(s) using board coordinates

Example:

```
setInPlane(1,1);
direction("");
move(100,200,false);
move(100,200,false);
doMove(100,200);

direction("h");
move(100,200,false);
move(100,200,false);
doMove(100,200);

direction("v");
move(100,200,false);
move(100,200,false);
doMove(100,200);
```

Parameters:

pt_x (X coordinate) Offset (vector) where to create the copy of the source objects

See also:

com.barco.ets.ucam.hypershell.HyperShell::doMove(Upoint)

com.barco.ets.ucam.hypershell.HyperShell::direction(String)

Parameters:

pt_y (Y coordinate) Offset (vector) where to create the copy of the source objects

See also:

com.barco.ets.ucam.hypershell.HyperShell::doMove(Upoint)

com.barco.ets.ucam.hypershell.HyperShell::direction(String)

Parameters:

bOnRefPoints If true, move is also applied to reference points

void move (Point *pt,*
boolean *bOnRefPoints*
)

Move (selected) object(s) using board coordinates

Example:

```
setInPlane(1,1);
direction("");
move(100,200,false);
move(100,200,false);
doMove(100,200);

direction("h");
move(100,200,false);
move(100,200,false);
doMove(100,200);

direction("v");
move(100,200,false);
move(100,200,false);
doMove(100,200);
```

Parameters:

pt Offset (vector) where to create the copy of the source objects

See also:

com.barco.ets.ucam.hypershell.HyperShell::doMove(Upoint)

com.barco.ets.ucam.hypershell.HyperShell::direction(String)

Parameters:

bOnRefPoints If true, move is also applied to reference points

void netlistBuild (String *target*)

Build Netlist

Parameters:

target Netlist target (job or layer) "job" - Builds netlist information for the whole job. All layers are used, whether they are active or not. Layers not yet in memory are loaded automatically. "layer" - Builds netlist information for the job. Only active layers are used.

void netlistClear ( )

Clear Netlist Removes all netlist information from a job.

void netlistReference (String *target*)

Create Netlist reference

Parameters:

target Netlist reference target (job or layer)

**void newJob (String *jobPath,*
String *jobName*
)**

Create a new job with the given name. The directory is automatically created if necessary. If the job already exists, a warning will be given and the existing job will be loaded.

Parameters:

jobPath The pathname for the job file
jobName The name of the job. The suffix .job will be omitted.

void notImplemented (String *sFuncName*)

Method warns that the HSH command does not exist

Parameters:

sFuncName function name

**void objAttribute (String *name,*
String *value*
)**

Sets the given value to the given object attribute. If the attribute exists, its value is changed to the new value. Otherwise the attribute is created. If the value is null the attribute with the given name is removed.

Parameters:

name attribute name
value attribute value

String objAttribute (String *name*)

Returns the value of the object attribute with given name.

Parameters:

name the object attribute name

Returns:

the value of the object attribute with given name or null if the attribute is not defined in the object or empty string if the value is not defined.

String objAttribute ( )

Returns comma separated list of the all object attributes. **Example:** ".out_scale=,nominal_y=yscale,.edgeline="

Returns:

comma separated list of the all object attributes.

Point objCenterPoint ( )

Returns the center point of the **Arc**

Returns:

Point the arc center point

double objClearance ( )

Measures clearance between two objects

Returns:

double value is clearance between two objects.

Rectangle objEnclosingBox ( )

Returns enclosing rectangle of the current object

Returns:

Rectangle enclosing rectangle of the current object

Point objFlash ( )

Returns flash point of the Flash or Vector Text

Returns:

Point the object flash point

Point objFromPoint ( )

Returns the first point of the Draw or **Arc**

Returns:

Point the first draw/arc point

String objInfo ( )

Prints information abou object

Returns:
Returns current object info

int objNet ( )

Returns object net number

Returns:
int net number

Point objPoint ( )

Returns flash point of Flash or VText objects or the first point of the Draw or **Arc**

Returns:
Point object flash point or the first draw/arc point

double objRing ( )

Measures inner clearance (ring) between two objects.

Returns:
A constant Not-a-Number (NaN) value when error, double value is clearance between two objects. Negative value means that second object is inside the first object.

void objSelect (String *sel*)

Selects/deselects the object according to given parameter.

Parameters:
sel "+" for select the object or "-" deselect the object

boolean objSelect ( )

Returns true if the object is selected.

Returns:
boolean true if the object is selected

String objSense ( )

Returns the sense of **Arc**

Returns:
String "cw" for ClockWise or "ccw" Counter-ClockWise, null if the Object is not **Arc**.

String objShape ( )

Returns shape of the object Aperture

Returns:
String possible values are "cir", "rec", "box", "oct", "con", "com", "the", "txt", "blo", "squ" and "don"

void objString (String *vtxString*)

Sets the String value of the VTX object

Parameters:
vtxString new string value in current VTX object

String objString ( )

Returns the String of the VTX object

Returns:
the String of the VTX object

Point objToPoint ( )

Returns the last (to) point of the Draw or **Arc**

Returns:
Point the last (to) draw/arc point

String objType ( )

Returns object type

Returns:
Possible values are "arc", "dra", "reg", "fla" or "vtx".

void offset (double *offset_x*, double *offset_y*)

Sets the Offset Used in Numbers Dialog

Parameters:
offset_x (X coordinate) the Offset

offset_y (Y coordinate) the Offset

void offset (Point *offset*)

Sets the Offset Used in Numbers Dialog

Parameters:

offset the Offset

Point offset ( )

Gets the Offset Used in Numbers Dialog

Returns:

Offset

void offsetX (double *offsetX*)

Sets the Offset x coordinate Used in Numbers Dialog

Parameters:

offsetX the Offset x coordinate

void offsetY (double *offsetY*)

Sets the Offset y coordinate Used in Numbers Dialog

Parameters:

offsetY the Offset y coordinate

void openAboutUcamco ( )

Opens ucamco website. (There is no closing function, since it is not strictly a dialog)

void openAboutUcamX ( )

Opens Ucam X product website. (There is no closing function, since it is not strictly a dialog)

void openAdvantools ( )

show Advantools

void openAMLIJobManager ( )

open AMLI Job Manager

void openAnamorphicScale ( )

show AnamorphicScale dialog

void openApeCreator ( )

open Aperture Creator

void openApeEditor ( )

open Aperture Editor

void openApertureAttributes ( )

show Aperture Attributes dialog

void openApertureManager ( )

show Aperture Manager dialog

void openAttributeEditor ( )

show Attribute Editor dialog

void openAttributeManager ( )

show Attribute Manager dialog

void openAutoDrill ( )

show AutoDrill dialog

void openAutoDrillEditor ( )

Show AutoDrill Editor

void openAutoFixture ( )

Show AutoFixture dialog

void openBarcode ( )

show Barcode dialog

void openBarcode128 ( )

show Barcode 128 dialog

void openBoardAnalyzer ( )

show Board Analyzer dialog

void openBoardSnapshot ( )

show Board Snapshot dialog

void openCalculatorSetup ( )

show Calculator Setup dialog

void openCamtek (String *sMachineCfg*)

show Camtek dialog

Parameters:

sMachineCfg

void openCFMEEOutput ( )

Open the CFMEE output dialog

void openCheckList ( )

show CheckList Dialog

void openCheckListDefineChecklist ( )

Show "CheckList: Define Checklist" Dialog

void openCheckListDefineSteps ( )

Show "CheckList: Define Steps" Dialog

void openClipping ( )

show Clipping dialog

void openColor ( )

Show Color dialog

void openConnect ( )

show Connect dialog

void openContourHandling ( )

show Contour Handling dialog

void openConvertAttributes ( )

show Attribute Converter dialog

void openCopperBalance ( )

open Copper Balance Dialog

void openCopperRepair ( )

show Copper Repair dialog

void openCoverlayOptimizer ( )

show Coverlay Optimizer dialog

void openCU9000Dialog ( )

Open DS DI output dialog

void openDatums ( )

show Datums dialog

void openDistort ( )

show Distort dialog

void openDraw (double *pt_x*,
double *pt_y*,
double *dis*
)

Open a draw. Inserts a break

Parameters:

pt_x (X coordinate) Open location
pt_y (Y coordinate) Open location
dis The clearance between the two endpoints of the open

void openDraw (Point *pt*,
double *dis*
)

Open a draw. Inserts a break

Parameters:

pt Open location
dis The clearance between the two endpoints of the open

void openDrawSlots ( )

show Draw Slots Dialog

void openDRC ( )

show DRC dialog

void openDrillInfo ( )

show Drill Info dialog

void openDrillMap ( )

open Drill Map Dialog

void openDrillOptimizer ( )

open Smart Drill Optimizer

void openDrillRoutSetups ( )

Show Drill/Rout Setups dialog

void openDrillTolerance ( )

Show Drill Tolerance dialog

void openDrillToolManager ( )

show Drill Tool Manager

void openDsAoi ( )

Open DS AOI dialog

void openDsAoiAdvanced ( )

Open DS AOI Advanced dialog (no closing function yet)

void openDSAOIialog ( )

Open DS AOI output dialog

void openDsAoiPreview ( )

Open DS AOI dialog

void openDsAoiQueue ( )

Open DS AOI Queue dialog (no closing function yet)

void openEditingToolbox ( )

show Editing Toolbox dialog

void openEditVectorText (double *pickPoint_x*,
double *pickPoint_y*
)

show Edit Vector Text dialog

Parameters:

pickPoint_x (X coordinate)
pickPoint_y (Y coordinate)

void openEditVectorText (Point *pickPoint*)

show Edit Vector Text dialog

Parameters:

pickPoint

void openErrors ( )

show Errors dialog

void openEtchCompensation ( )

show Etch Compensation dialog

void openExpand ( )

show Expand dialog

void openExternalLinkManager ( )

show External Link Manager

void openFiducials ( )

show Fiducials dialog

void openFillAngledPattern ( )

open Fill Angled Pattern Dialog

void openFillPattern ( )

open Fill Pattern Dialog

void openFillVector ( )

open Fill Vector Dialog

void openFlashMaker ( )

show FlashMaker dialog

void openFlexManager ( )

open uFlex Manager

void openFlipJob ( )

show Flip Job dialog

void openFrame (String *sFrameName*)

Opens custom dockable frame with given identification

Parameters:

sFrameName identification frame given by getFrameID() method of CustomFrame class

void openGridParameters ( )

show Grid Parameters dialog

void openHelp ( )

Opens the application's documentation. (There is no closing function, since it is not strictly a dialog)

void openHelpOnHelp ( )

Opens help documentation. (There is no closing function, since it is not strictly a dialog)

void openHelpOnHypertool ( )

Opens hypertool documentation. (There is no closing function, since it is not strictly a dialog)

void openHelpOnResources ( )

Opens resource's documentation. (There is no closing function, since it is not strictly a dialog)

void openHelpOnVersion ( )

Opens version documentation. (There is no closing function, since it is not strictly a dialog)

void openHiPot ( )

show HiPot dialog

void openImageCompare ( )

show Image Compare dialog

void openImpedanceControl ( )

show Impedance Control dialog

void openImportIPC356 ( )

show Import IPC356 dialog (no closing command)

void openImportMET ( )

show Import MET dialog (no closing command)

void openImportODBxx ( )

show Import ODBxx Steps dialog

void openImportWF ( )

show Import WF dialog (no closing command)

void openInsertContourText ( )

show Insert Contour Text dialog

void openInsertVectorText ( )

show Insert Vector Text dialog

void openJob (String *jobName*)

Open the job with the given name. In case the job does not exist, a warning will be given and an empty new job will be loaded.

Parameters:

jobName The full name of the job including the path.

void openJob_shm (String *sShmName*)

read job from shared memory

Parameters:

sShmName - Name of the shared memory

void openJobCreate ( )

Show create job dialog (no close)

void openJobDefinition ( )

show Job Definition dialog

void openJobEdit ( )

show Job Edit dialog

void openJobEditor ( )

Show Job Editor dialog

void openJobEditorOptions ( )

Show Job Editor Options dialog

void openJobLoad ( )

Show open job dialog (no close)

void openJobMerge ( )

show Job Merge dialog

void openJobPlaneSetup ( )

Show Job Plane Setup dialog

void openJobPrint ( )

Show Job Print dialog

void openJobView ( )

show Job View dialog

void openLayerEdit ( )

opens Layer Modify dialog

void openLegendOptimizer ( )

show Legend Optimizer dialog

void openLicenseHelp ( )

Opens license agreement help. (There is no closing function, since it is not strictly a dialog)

void openLoadCheckList ( )

show Load CheckList Dialog

void openMagnifier ( )

show Magnifier window

void openMarkupAssistant ( )

open Markup Assistant

void openMessages ( )

show Messages log window

void openMLIOutput ( )

open MLI Output dialog

void openModels ( )

show Models dialog

void openNetCompare ( )

show Net Compare dialog

void openNetfixSetup ( )

show UTest dialog

void openNonFunctionalPad ( )

show Non-Functional pads dialog

void openNumbers ( )

show Numbers dialog

void openObjectAttributes ( )

show Object Attributes dialog

void openObjectCompare ( )

show Object Compare dialog

void openOutputAccumatch ( )

Open Accumatch Output dialog

void openOutputAOI ( )

Open AOI Output dialog

void openOutputCAD ( )

show Output CAD dialog

void openOutputCamtek ( )

Open Camtek Output dialog

void openOutputDrillRout (String *sDrillMachine*)

show Output Drill/Rout dialog

Parameters:

sDrillMachine

void openOutputDsDi ( )

Open DS DI output dialog

void openOutputDsDiPreview ( )

Open DS DI Preview dialog

void openOutputDsDiQueue ( )

Open DS DI queue dialog (no closing function yet)

void openOutputNetlist ( )

show Output Netlist dialog

void openOutputOrbot ( )

Open Orbot Output dialog

void openOutputSapphire ( )

Open Sapphire Output dialog

void openOutputScoring ( )

show Output Scoring dialog

void openOutputSmartArgos ( )

Open SmartArgos Output dialog

void openOutputTrackscan ( )

Open Trackscan Output dialog

void openOutputUxpAutomanager ( )

Open Output UXP Automanager dialog

void openOutputUxpEtec ( )

Open Output UXP Etec dialog

void openOutputUxpLocal ( )

Open Output UXP Local dialog (no closing function yet)

void openPanelFramesCoupons ( )

Show Panel Frames Coupons dialog

void openPanelLinks ( )

Show Panel Links dialog

void openPanelPlus ( )

show PanelPlus dialog

void openPanelReproduce ( )

show Panel Reproduce dialog

void openPanelSetup ( )

Show Panel Setup dialog

void openPanelStepRepeat ( )

show Panel Step Repeat dialog

void openPlaneAdjuster ( )

show Plane Adjuster dialog

void openPlotParameters ( )

show Plot Parameters dialog

void openPPMonitor ( )

show PPMonitor dialog

void openProductionStagesEditor ( )

show Production Stages Editor dialog

void openQueryNet ( )

show Query net dialog

void openQueryObject ( )

show Query Object dialog

void openReferencePoints ( )

show Reference Points dialog

void openRegister ( )

show Register dialog

void openRemoveAttributes ( )

show Remove Attributes dialog

void openRepair (String *szLabname*)

Show Repair dialog

Parameters:

szLabname

void openRoutManager ( )

show Rout Manager dialog

void openRoutManagerCleanUp ( )

show Rout Manager dialog on Clean Up page

void openRoutManagerDimensioning ( )

show Rout Manager dialog on Dimensioning page

void openRoutManagerEditor ( )

show Rout Manager dialog on Editor page

void openRoutManagerTools ( )

show Rout Manager dialog on Tools page

void openSaveJobAs ( )

Open Save Current job as...

void openSaveLayout ( )

show Save Window Layout dialog

void openSecureEtchCompensation ( )

show Secure Etch Compensation dialog

void openSelections ( )

show Selections dialog

void openSetupOptions ( )

Show Setup Options dialog

void openSetupSave ( )

Show Save dialog

void openShavePads ( )

show Shave Pads dialog

void openSignalLayerAdjuster ( )

show Signal Layer Adjuster dialog

void openSignalLayerAdjusterAssistant ( )

show Signal Layer Adjuster Assistant dialog

void openSilkOptimizer ( )

open Silk Optimizer

void openSmartCamtek (String *sMachineCfg*)

show SmartCamtek dialog

Parameters:

sMachineCfg

void openSmartDRC ( )

show Smart DRC dialog

void openSmartFix ( )

open SmartFix dialog

void openSmartplot ( )

Open Smartplot dialog

void openSmartSR ( )

Show Smart S&R dialog

void openSmartStart ( )

show Smart Start dialog

void openSoldermask ( )

open Soldermask Dialog

void openSoldermaskOptimizer ( )

open Soldermask Optimizer Dialog

void openTearDrop ( )

open Tear Drop Dialog

void openTechnicalAnalyzer ( )

show Technical Analyzer dialog

void openTestpointEdit ( )

show Testpoint edit dialog

void openToolbarManager ( )

show Toolbar Manager Dialog

void openToolbars ( )

show Toolbars Dialog

void openTransformObjects ( )

show Transform Objects dialog

void openTransformObjectsBGAPads ( )

show Transform Objects BGA Pads dialog

void openTransformObjectsBGATracks ( )

show Transform Objects BGA Tracks dialog

void openTransformObjectsEdit ( )

show Transform Objects Edit dialog

void openTransformObjectsRescale ( )

show Transform Objects Rescale dialog

void openUcamDbEditor ( )

Show Ucamdb Editor dialog

void openUndoRedoDetails ( )

show Undo/Redo Details

void openUTest ( )

show UTest dialog

void openUtestUtilities ( )

Show Utest Utilities dialog

void openValidateLayer ( )

show Layer Validation dialog (or not, if everything is fine)

void openVectorTextFont ( )

show Vector Text Font dialog

void openVerifyArcsDraws ( )

show Verify Arcs and Draws dialog

void openViewGuide ( )

show View Guide dialog

void openYsphotechOutput ( )

Open the Ysphotech output dialog

void optimizeDrill (int *nPasses*,
int *optMode*,
**double *yxTime*
)**

Tools->Tooling->SmartDrill Optimize

Parameters:

nPasses - Number of Passes
optMode - Optimize Method: Path Length or Drill Time
yxTime - Time Ratio Y/X, used for Drill Time Optimize Method

void optimizeMaskLayer (double *dMinRing*,
double *dMaxRing*,
double *dMaskToCopper*,
double *dMaskToMask*,
double *dBigRing*
)

Optimize Soldermask layer

Parameters:

dMinRing - The minimum value accepted by the ring for adding mask to the copper pad.
dMaxRing - The value of the ring around the copper pad which is used to start searching for solder mask pad.
dMaskToCopper - The minimum clearance that is still legal between a pad of the mask layer and copper on the copper layer.
dMaskToMask - The minimum clearance that is still permissible between two pads of the mask layer.
dBigRing - Big Pad Ring

String osChDir ( )

Returns current directory

Returns:

null if the function failed, otherwise full path to the current directory.

String osChDir (String *sDir*)

Sets the current directory.

Parameters:

sDir The directory specification.

Returns:

null if the function failed, otherwise the full path to the new current directory.

**int osCopy (String *sSrcName,*
String *sDstName*
)**

Copies a file to another file.

Parameters:

sSrcName The specification of the file to copy.

sDstName The specification of the destination file.

Returns:

status 0 if copy is OK

String osCreateTmpDir (String *sBasePath*)

Function that creates tmp directory under given path and returns its name **Warning:** All temporary files and files placed into temporary directories created during a script are deleted at the end of the script run.

Parameters:

sBasePath the base path where the temporary directory will be created.

Returns:

null if the function failed; otherwise the full path to the new temporary directory.

Exceptions:

IOException

String osCreateTmpDir ( )

Function creates tmp directory under system $TEMP directory and returns its name **Warning:** All temporary files and files placed into temporary directories created during a script are deleted at the end of the script run.

Returns:

null if the function failed; otherwise the full path to the new temporary directory.

Exceptions:

IOException

int osDelete (String *sFileName*)

Delete specified file

Parameters:

sFileName file name that will be deleted

Returns:

status 0 if delete is OK

ObjectList osFileInfo (String *sPath*)

Returns objectlist of the file information

Parameters:

sPath The full path of the file we need get the information

Returns:

objectlist with the file information

Example:

```
 list = osGetFileList("D:\\ucam\\ucam\\custom", true);

 item = forEachItem(list) {
     fileInfo = osFileInfo(item);
     fileInfo = osFileInfo(item);

     if (isFile(fileInfo)) {
         print("FILE: " + getFileName(fileInfo) + " (size " + getFileSize(fileInfo)
+ "b)");
     }
     else if (isDirectory(fileInfo)) {
         print("DIRECTORY: " + getFileName(fileInfo));
     }
     print("\tat location " + getFileParent(fileInfo));
     if (canRead(fileInfo) && !canWrite(fileInfo)) {
         print("\tis READ-ONLY");
     }
     if (isHidden(fileInfo)) {
         print("\tis HIDDEN");
     }
     print("\twas last modified on " + getFileLastModified(fileInfo));
 }
```

Output:

```
 FILE: Udualstrip.java (size 4808b)
     at location D:\\ucam\\ucam\\custom\\impedance
     is HIDDEN
     was last modified on Thu Aug 11 12:39:19 CEST 2011
 FILE: Uembmstrip.java (size 3951b)
     at location D:\\ucam\\ucam\\custom\\impedance
     was last modified on Tue Nov 01 12:20:51 CET 2005
 FILE: Usrfmstrip.java (size 3993b)
     at location D:\\ucam\\ucam\\custom\\impedance
     was last modified on Tue Nov 01 12:20:51 CET 2005
 FILE: Usymstrip.java (size 4036b)
     at location D:\\ucam\\ucam\\custom\\impedance
     is READ-ONLY
     is HIDDEN
     was last modified on Tue Nov 01 12:20:51 CET 2005
```

See also:

canRead(ObjectList)

canWrite(ObjectList)

getFileLastModified(ObjectList)

getFileName(ObjectList)

getFileParent(ObjectList)

getFileSize(ObjectList)

isDirectory(ObjectList)

isFile(ObjectList)

isHidden(ObjectList)

**ObjectList osGetFileList (String *sDir,*
String *sFileMask,*
boolean *bRecurse,*
boolean *bFullPath,*
boolean *bWithDirs*
)**

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir	The directory path
sFileMask	file mask filter supports WildCards (e.g. "*.vhs", "?_script.*")
bRecurse	if `true` the function goes to subdirectories as well
bFullPath	if `true` the function returns the list of the full pathes
bWithDirs	if `true` the OjectList contains also directories

Returns:

Object List containing all file names in given directory and subdirectories if bRecurse is set to true and matching given sFileMask. The ObjectList includes the names of the directories if bWithDirs is set to `true`.

Exceptions:

IOException	If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

**ObjectList osGetFileList (String *sDir,*
String *sFileMask,*
boolean *bRecurse,*
boolean *bFullPath*
)**

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir	The directory path
sFileMask	file mask filter supports WildCards (e.g. "*.vhs", "?_script.*")
bRecurse	if `true` the function goes to subdirectories as well
bFullPath	if `true` the function returns the list of the full paths

Returns:

Object List containing all file names in given directory and subdirectories if bRecurse is set to true and matching given sFileMask.

Exceptions:

IOException	If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

**ObjectList osGetFileList (String *sDir,*
String *sFileMask,*
boolean *bRecurse*
)**

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir The directory path
sFileMask file mask filter supports WildCards (e.g. "*.vhs", "?_script.*")
bRecurse if true the function goes to subdirectories as well

Returns:

Object List containing all file and directory full paths in given directory and subdirectories if bRecurse is set to `true` and matching given sFileMask.

Exceptions:

IOException If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

**ObjectList osGetFileList (String *sDir,*
String *sFileMask*
)**

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir The directory path
sFileMask file mask filter supports WildCards (e.g. "*.vhs", "?_script.*")

Returns:

Object List containing all file and directory full paths in given directory matching the sFileMask. It is not recursive.

Exceptions:

IOException If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

**ObjectList osGetFileList (String *sDir,*
boolean *bRecurse*
)**

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir The directory path
bRecurse if `true` the function goes to subdirectories as well

Returns:

Object List containing all file names in given directory and subdirectories if bRecurse is set to true.

Exceptions:

IOException If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

**ObjectList osGetFileList (String *sDir,*
boolean *bRecurse,*
boolean *bFullPath,*
boolean *bWithDirs*
)**

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir The directory path
bRecurse if `true` the function goes to subdirectories as well
bFullPath if `true` the function returns the list of the full pathes
bWithDirs if `true` the OjectList contains also directories

Returns:

Object List containing all file names in given directory

Exceptions:

IOException If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

ObjectList osGetFileList (String *sDir*)

Enumerates directory and returns ObjectList of filenames (string)

Parameters:

sDir The directory path

Returns:

Object List containing all file and directory full paths in given directory. It is not recursive.

Exceptions:

IOException If an I/O error occurs, which is possible because the construction of the file pathname list may require filesystem queries

int osMarkAsTmp (String *sName*)

Mark file (or directory) as temporary. Each marked file/directory will be deleted on the end of script automatically!

Parameters:

sName file/directory with the name is marked as temporary

Returns:

0 if file is successfuly marked as temporary; otherwise returns 1

Exceptions:

EvalError

int osMkDir (String *sDirName*)

Creates the directory named by the given pathname.

Parameters:

sDirName Directory path that will be created

Returns:

0 if and only if the directory was created; 1 otherwise

**int osMove (String *sSrcName,*
String *sDstName*
)**

Renames/Moves a file.

Parameters:

sSrcName The name of the file to rename.
sDstName The new name for the file.

Returns:

status 0 if remane/move is OK

void osRmDir (String *sDirName*)

Remove empty directory

Parameters:

sDirName Directory name

void osRmTree (String *sDirName*)

Remove non-empty directory/subtree

Parameters:

sDirName Directory name

**int osUnTgz (String *sTgzArchive,*
String *sDstDir*
)**

This function creates a subdirectory with the basename of the tgz-file and will gunzip and untar the given tgz-archive under this subdirectory.

Parameters:

sTgzArchive TGZ archive
sDstDir Destination directory

Returns:

0 if the given file is successfuly decompressed or 1 if it failed

int osUnZip (String *sZipArchive,*
String *sDstDir*
)

Uncompress .zip file to defined destination directory

Parameters:

sZipArchive ZIP archive
sDstDir Destination directory

Returns:

0 if the given file is successfuly decompressed or 1 if it failed

void outAtgFixture (String *key,*
String *sTool,*
String *iRes,*
int *iSession*
)

Converts all active layers of this job to the Atg Fixture format.

Parameters:

key - language anf, anfref, anf_a2000, anfref_a2000
sTool - tool file
iRes - resource file
iSession - session id

void output274x (String *sRes*)

Converts all active layers of this job to the Gerber 274x format.

Parameters:

sRes path to the resource file

void outputAft (String *res*)

Converts all active layers of this job to the AFT format.

Parameters:

res path to the resource file

void outputAoi (boolean *bCadData,*
boolean *bReference*
)

Generate AOI output

Parameters:

bCadData - generate Cad info (LP file)

bReference - generate Reference data (GER file)

void outputAtf ( )

Converts all active layers of this job to the ATF format.

ObjectList outputAutoDrill (String *sDrjFile*)

Output AutoDrill

Parameters:

sDrjFile AutoDrill configuration file path.

Returns:

ObjectArray, with the names of the output file names.

void outputCFMEE (String *outputPath*,
boolean *reverse*,
double *marginx*,
double *marginy*,
boolean *distort*,
double *distortx*,
double *distorty*,
double *resizex*,
double *resizey*,
boolean *deleteOutside*
)

Generate CFMEE

Parameters:

outputPath	the location where the gdsii output will be written
reverse	whether the image should be reversed
marginx	margin x when reversing the image
marginy	margin y when reversing the image
distort	
distortx	
distorty	
resizex	anamorphic resize x value
resizey	anamorphic resize y value
deleteOutside	whether to delete objects outside the outline

void outputCli ( )

Converts all active layers of this job to the CLI format.

void outputColorPDF (String *sPdfFullPath*)

Create a pdf file from given layer (active in plane 1 - red layer) Apertures will be coloring by attribute uColor

Parameters:

sPdfFullPath full path to new PDF file, include file name

void outputDp40 (double *pt_x*,
double *pt_y*,
boolean *bPositive*,
boolean *bMirrorx*,
boolean *bMirrory*,
double *dLaserPower*,
int *iPolygonSpeed*,
int *iPcbFormat*,
String *unit*
)

Parameters:

pt_x (X coordinate)
pt_y (Y coordinate)
bPositive
bMirrorx
bMirrory
dLaserPower
iPolygonSpeed
iPcbFormat
unit

void outputDp40 (Point *pt*,
boolean *bPositive*,
boolean *bMirrorx*,
boolean *bMirrory*,
double *dLaserPower*,
int *iPolygonSpeed*,
int *iPcbFormat*,
String *unit*
)

Parameters:

pt
bPositive
bMirrorx
bMirrory
dLaserPower
iPolygonSpeed
iPcbFormat
unit

```
void outputDxf ( String  unit,
                 int     iConturize,
                 int     iKeepTXT,
                 double  dExpandArcs,
                 int     iCenterLine,
                 int     iAllInOne
               )
```

Converts all active layers of this job to the DXF format.

Parameters:

unit - units
iConturize - contourize
iKeepTXT - keep TXT
dExpandArcs - expand arcs
iCenterLine - center line
iAllInOne - all in one

```
void outputDxfV6 ( String  unit )
```

Converts all active layers of this job to the DXF v6 format.

Parameters:

unit - units

```
void outputEie ( String      sRes,
                 ObjectList  par
               )
```

Converts all active layers of this job to the EIE format.

Parameters:

sRes path to the resource file
par array of parameters

```
void outputEtec ( String   sResistOuter,
                  String   sResistInner,
                  double   mediaX,
                  double   mediaY,
                  int      iAlignType,
                  int      iLevelType,
                  int      iCycles,
                  String   sDate,
                  String   sTime,
                  boolean  bAuto,
                  double   dScaleX,
                  double   dScaleY,
                  double   dScaleOriX,
```

```
                double  dScaleOriY,
                String  sMDFfile,
                String  sResource
              )
```

Generates ETEC output

Parameters:

sResistOuter
sResistInner
mediaX
mediaY
iAlignType
iLevelType
iCycles
sDate
sTime
bAuto
dScaleX
dScaleY
dScaleOriX
dScaleOriY
sMDFfile
sResource

```
void outputExt ( String      lan,
                 String      too,
                 String      res,
                 ObjectList  resdb,
                 String      inc1,
                 String      inc2,
                 int         session,
                 Object[]    pre,
                 Object[]    pos,
                 Object      notUsed
               )
```

Converts all active layers of this job to many format types.

Parameters:

lan - output language
too - tool file
res - resource file
resdb - resource database
inc1 - part of Outpar.inc1
inc2 - part of Outpar.inc2
session - part of Outpar.session
pre - part of Outpar.pre
pos - part of Outpar.pos
notUsed - the parameter is not used at all must be set to `null`

```
void outputExt ( String     lan,
                 String     too,
                 String     res,
                 ObjectList resdb,
                 String     inc1,
                 String     inc2,
                 int        session,
                 Object[]   pre,
                 Object[]   pos
               )
```

Converts all active layers of this job to many format types.

Parameters:

lan - output language
too - tool file
res - resource file
resdb - resource database
inc1 - part of Outpar.inc1
inc2 - part of Outpar.inc2
session - part of Outpar.session
pre - part of Outpar.pre
pos - part of Outpar.pos

```
void outputHimt ( double datum_x,
                  double datum_y,
                  double offset_x,
                  double offset_y,
                  String mirror,
                  String rotation
                )
```

Converts all active layers of this job to the HIMT format.

Parameters:

datum_x (X coordinate) - datum point
datum_y (Y coordinate) - datum point
offset_x (X coordinate) - offset point
offset_y (Y coordinate) - offset point
mirror - mirror
rotation - rotation

```
void outputHimt ( Point  datum,
                  Point  offset,
                  String mirror,
                  String rotation
                )
```

Converts all active layers of this job to the HIMT format.

Parameters:

datum - datum point
offset - offset point
mirror - mirror
rotation - rotation

void outputIpc2581 ( )

Converts all active layers of this job to the IPC2581 format.

void outputIpcUfd (String *key*,
String *res*,
String *version*
)

Converts all active layers of this job to the IPC350, IPC356, MET* and MNF* formats.

Parameters:
key - language
res - resource file
version - Ucam version

void outputLpg (int *iPpi*,
int *iChoke*,
double *offset_x*,
double *offset_y*
)

Parameters:
iPpi - resolution
iChoke - choke
offset_x (X coordinate)
offset_y (Y coordinate)

void outputLpg (int *iPpi*,
int *iChoke*,
Point *offset*
)

Parameters:
iPpi - resolution
iChoke - choke
offset

int outputManiaSapphire (String *sOutputPath*,
String *sDescription*,

```
                    String    sGeometryfile,
                    boolean   bStatistics,
                    boolean   bDrill
                )
```

Generates Mania Sapphire output

Parameters:

sOutputPath - output path
sDescription - description
sGeometryfile - geometry file
bStatistics - statistic
bDrill - drill

Returns:

error status - 0 means OK

```
void outputMda ( String     sPath,
                 ObjectList par,
                 Object[]   subpar,
                 int        iApr,
                 int        iSubfig,
                 int        iRenum
               )
```

Converts all active layers of this job to the MDA format.

Parameters:

sPath path.
par G04 parameters for the main MDA file.
subpar G04 parameters for the sub MDA file.
iApr The aperture scale factor.
iSubfig Subfigures allowed when 1.
iRenum Renumber apetures when 1 (default 0).

```
void outputNec ( String too )
```

Converts all active layers of this job to the NEC format.

Parameters:

too - tools file

```
void outputOdbxx ( String res )
```

Converts all active layers of this job to the ODB++ format.

Parameters:

res - resource file

void outputOdbxxv7 (String *res*)

Converts all active layers of this job to the ODB++ v7 format.

Parameters:

res - resource file

void outputOif (String *oifVersion*,
int *byJob*,
int *pan*,
int *fillin*
)

Converts all active layers of this job to the OI2002 or OI5000 format.

Parameters:

oifVersion - language oi2002 or oi5000
byJob - value of "uout_oifbac_tog"
pan - value of "uout_oifpan_tog"
fillin - value of "uout_oif_fil"

boolean outputOrbot ( )

Output Orbot - should be extended or canceled

Returns:

status

void outputPdf ( )

Converts all active layers of this job to the PDF format.

void outputProbe (String *sLang*,
int *iSession*,
int *iAccuracy*
)

Converts all active layers of this job to the flying probe formats probot and rislang.

Parameters:

sLang The language : "probot" or "rislang".
iSession The session number.
iAccuracy Accuracy for probot.

int outputRaid ( )

Converts all active layers of this job to the RAID format.

Returns:
status

```
void outputRpd ( int    iPpi,
                 double datum_x,
                 double datum_y,
                 double offset_x,
                 double offset_y,
                 String sMirror,
                 String sRotation
               )
```

Converts all active layers of this job to the RPD format.

Parameters:

iPpi
datum_x (X coordinate)
datum_y (Y coordinate)
offset_x (X coordinate)
offset_y (Y coordinate)
sMirror
sRotation

```
void outputRpd ( int    iPpi,
                 Point  datum,
                 Point  offset,
                 String sMirror,
                 String sRotation
               )
```

Converts all active layers of this job to the RPD format.

Parameters:

iPpi
datum
offset
sMirror
sRotation

```
boolean outputSchmid ( int    resolution,
                       double maskRectangle_xmin,
                       double maskRectangle_ymin,
                       double maskRectangle_xmax,
                       double maskRectangle_ymax,
                       int    maskRotation,
                       String maskMirror,
                       String maskPolarity,
                       int    equipmentRotation,
```

```
                    String      equipmentMirror,
                    double      offsetX,
                    double      offsetY,
                    ObjectList  fiducials,
                    String      imagePath,
                    String      configPath,
                    String      batchFile
                )
```

Generates Schmid output (tiff and xml file) for all active layers of the current job.

Parameters:

resolution	The resolution of the tiff file.
maskRectangle_xmin	(left boundary of rectangle) Describes the enclosing rectangle of the tiff file.
maskRectangle_ymin	(bottom boundary of rectangle) Describes the enclosing rectangle of the tiff file.
maskRectangle_xmax	(right boundary of rectangle) Describes the enclosing rectangle of the tiff file.
maskRectangle_ymax	(top boundary of rectangle) Describes the enclosing rectangle of the tiff file.
maskRotation	The rotation of the tiff file (in degrees).
maskMirror	The mirror setting of the tiff file ("", "X", "Y" or "XY").
maskPolarity	The polarity of the tiff file ("P" or "N").
equipmentRotation	The rotation of the tiff file on the inkjet equipment.
equipmentMirror	The mirroring of the tiff file on the inkjet equipment.
offsetX	The X offset of the tiff image on the inkjet equipment.
offsetY	The Y offset of the tiff image on the inkjet equipment.
fiducials	List of fiducial points (in CAD coordinates).
imagePath	The directory where to store the image file.
configPath	The directory where to store the xml file.
batchFile	Specifies a postprocessing .bat file which should be run when the output is created. The specification can contain i to denote the image file and x to denote the xml file.

Returns:

output return value

```
boolean outputSchmid ( int         resolution,
                       Rectangle   maskRectangle,
                       int         maskRotation,
                       String      maskMirror,
                       String      maskPolarity,
                       int         equipmentRotation,
                       String      equipmentMirror,
                       double      offsetX,
                       double      offsetY,
                       ObjectList  fiducials,
                       String      imagePath,
                       String      configPath,
                       String      batchFile
                     )
```

Generates Schmid output (tiff and xml file) for all active layers of the current job.

Parameters:

resolution	The resolution of the tiff file.

maskRectangle	Describes the enclosing rectangle of the tiff file.
maskRotation	The rotation of the tiff file (in degrees).
maskMirror	The mirror setting of the tiff file ("", "X", "Y" or "XY").
maskPolarity	The polarity of the tiff file ("P" or "N").
equipmentRotation	The rotation of the tiff file on the inkjet equipment.
equipmentMirror	The mirroring of the tiff file on the inkjet equipment.
offsetX	The X offset of the tiff image on the inkjet equipment.
offsetY	The Y offset of the tiff image on the inkjet equipment.
fiducials	List of fiducial points (in CAD coordinates).
imagePath	The directory where to store the image file.
configPath	The directory where to store the xml file.
batchFile	Specifies a postprocessing .bat file which should be run when the output is created. The specification can contain i to denote the image file and x to denote the xml file.

Returns:

output return value

```
void outputSl13 ( String sResources,
                  String sKey
                )
```

Converts all active layers of this job to the SL3.7 or SL13.9 format.

Parameters:

sResources - resource file path
sKey - language

```
void outputSprint ( String  sOutputFolder,
                    boolean bStandardMark,
                    String  sCopperName,
                    String  sRefName,
                    String  sAttributeZero,
                    int     iZeroPointNumber,
                    String  sAttributeCamera,
                    int     iCameraNumber,
                    int     iRotation0,
                    int     iRotation90,
                    int     iRotation180,
                    int     iRotation270,
                    String  sText1,
                    String  sText2,
                    String  sText3,
                    String  sCleanOption
                  )
```

Generates Sprint output for the current job.

Parameters:

sOutputFolder	The output folder
bStandardMark	

sCopperName
sRefName
sAttributeZero
iZeroPointNumber
sAttributeCamera
iCameraNumber
iRotation0
iRotation90
iRotation180
iRotation270
sText1
sText2
sText3
sCleanOption

void outputSys ( )

Converts all active layers of this job to the Systronic format.

void outputTiff (String *sPath,*
String *sExt,*
String *sOptions,*
int *iResolution*
)

Makes a pixel file of the job/layer/aperture.

Parameters:

sPath	The path of the file.
sExt	The extension of the file.
sOptions	
iResolution	The output resolution in ppi.

int outputTs3 (boolean *bDrilledBoards,*
double *dSpace*
)

Parameters:

bDrilledBoards	True to output the drill layers too. False otherwise.
dSpace	The nominal space.

Returns:

output return value

void outputWf2 (ObjectList *par*)

Converts all active layers of this job to the WF2 format.

Parameters:

par - array with all parameters

void outputXdpf (String *sPath*)

Generates XDPF job

Parameters:

sPath - destination path

```
void outputYsphotech ( boolean  imagecomp,
                       String   outputPath,
                       String   reverse,
                       double   marginx,
                       double   marginy,
                       boolean  mirrorx,
                       boolean  mirrory,
                       double   rotate,
                       double   distortx,
                       double   distorty,
                       double   resize,
                       boolean  keepArrays,
                       boolean  deleteOutside,
                       boolean  autoDetected,
                       String   layername
                     )
```

Generate ysphotech output on layer

Parameters:

imagecomp	
outputPath	the location where the gdsii output will be written
reverse	Yes, No or False. Yes will reverse the image, No won't reverse the image. And False will prepare to reverse later: add $$ to the name and inverse the resize value
marginx	margin x when reversing the image
marginy	margin y when reversing the image
mirrorx	whether the image should be mirrored along the x-axis
mirrory	whether the image should be mirrored along the y-axis
rotate	rotation angle in degrees
distortx	distort x value
distorty	distort y value
resize	resize value
keepArrays	whether to keep the array structure
deleteOutside	whether to delete objects outside the outline
autoDetected	whether or not alignment points are automatically detected or selected by the user
layername	the name of the layer to be outputted, GDSII output will only happen when the layer is active

```
void outputYsphotech ( String   outputPath,
```

```
                String  reverse,
                double  marginx,
                double  marginy,
                boolean mirrorx,
                boolean mirrory,
                double  rotate,
                double  distortx,
                double  distorty,
                double  resize,
                boolean keepArrays,
                boolean deleteOutside,
                boolean autoDetected,
                String  layername
            )
```

Generate ysphotech output on all active layers

Parameters:

outputPath	the location where the gdsii output will be written
reverse	
marginx	margin x when reversing the image
marginy	margin y when reversing the image
mirrorx	whether the image should be mirrored along the x-axis
mirrory	whether the image should be mirrored along the y-axis
rotate	rotation angle in degrees
distortx	distort x value
distorty	distort y value
resize	resize value
keepArrays	whether to keep the array structure
deleteOutside	whether to delete objects outside the outline
autoDetected	whether or not alignment points are automatically detected or selected by the user
layername	the name of the layer to be outputted, GDSII output will only happen when the layer is active

```
void pajPlaneAdjust ( double  dPlatedClearance,
                      double  dUnplatedClearance,
                      double  dRingSize,
                      double  dRingClearance,
                      double  dLineWidth,
                      double  dCuClearance,
                      boolean bDoCut,
                      double  dOutlineCLearance
                    )
```

Move word with given uText value over distance dx, dy, unless the distance is larger than limit and enforcelimit is true

Parameters:

dPlatedClearance
dUnplatedClearance
dRingSize
dRingClearance

dLineWidth
dCuClearance
bDoCut
dOutlineCLearance

void pajPlaneAdjust (**double** ***dPlatedClearance,***
double ***dUnplatedClearance,***
double ***dRingSize,***
double ***dRingClearance,***
double ***dLineWidth,***
double ***dCuClearance,***
boolean ***bDoCut,***
double ***dOutlineClearance,***
boolean ***bOutputAsContour,***
boolean ***bSaveBackup,***
boolean ***bErrorPopups***
)

Move word with given uText value over distance dx, dy, unless the distance is larger than limit and enforcelimit is true

Parameters:

dPlatedClearance
dUnplatedClearance
dRingSize
dRingClearance
dLineWidth
dCuClearance
bDoCut
dOutlineClearance
bOutputAsContour
bSaveBackup
bErrorPopups

void pajPlaneAdjust (**double** ***dPlatedClearance,***
double ***dUnplatedClearance,***
double ***dRingSize,***
double ***dRingClearance,***
double ***dLineWidth,***
double ***dCuClearance,***
boolean ***bDoCut,***
double ***dOutlineClearance,***
boolean ***bOutputAsContour,***
boolean ***bSaveBackup***
)

Move word with given uText value over distance dx, dy, unless the distance is larger than limit and enforcelimit is true

Parameters:

dPlatedClearance
dUnplatedClearance
dRingSize
dRingClearance
dLineWidth
dCuClearance
bDoCut
dOutlineClearance
bOutputAsContour
bSaveBackup

void panelStepRepeat (**double** ***pStart_x,***
double ***pStart_y,***
int ***iRepeatX,***
int ***iRepeatY,***
double ***dStepX,***
double ***dStepY,***
String ***sFlashPoint***
)

Step Repeat Panel

Parameters:

pStart_x	(X coordinate) Position of first flashPoint
pStart_y	(Y coordinate) Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction
sFlashPoint	Flashpoint "middle", "zero" or "center"

void panelStepRepeat (**Point** ***pStart,***
int ***iRepeatX,***
int ***iRepeatY,***
double ***dStepX,***
double ***dStepY,***
String ***sFlashPoint***
)

Step Repeat Panel

Parameters:

pStart	Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction
sFlashPoint	Flashpoint "middle", "zero" or "center"

```
void panelStepRepeatCenter ( double pStart_x,
                             double pStart_y,
                             int    iRepeatX,
                             int    iRepeatY,
                             double dStepX,
                             double dStepY
                           )
```

Step Repeat Panel Flash point = Center

Parameters:

pStart_x	(X coordinate) Position of first flashPoint
pStart_y	(Y coordinate) Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction

```
void panelStepRepeatCenter ( Point  pStart,
                             int    iRepeatX,
                             int    iRepeatY,
                             double dStepX,
                             double dStepY
                           )
```

Step Repeat Panel Flash point = Center

Parameters:

pStart	Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction

```
void panelStepRepeatJobZero ( double pStart_x,
                              double pStart_y,
                              int    iRepeatX,
                              int    iRepeatY,
                              double dStepX,
                              double dStepY
                            )
```

Step Repeat Panel Flash point = Job Zero

Parameters:

pStart_x	(X coordinate) Position of first flashPoint
pStart_y	(Y coordinate) Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction

```
void panelStepRepeatJobZero ( Point  pStart,
                              int    iRepeatX,
                              int    iRepeatY,
                              double dStepX,
                              double dStepY
                            )
```

Step Repeat Panel Flash point = Job Zero

Parameters:

pStart	Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction

```
void panelStepRepeatMiddle ( double pStart_x,
                             double pStart_y,
                             int    iRepeatX,
                             int    iRepeatY,
                             double dStepX,
                             double dStepY
                           )
```

Step Repeat Panel Flash point = Middle

Parameters:

pStart_x	(X coordinate) Position of first flashPoint
pStart_y	(Y coordinate) Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction
dStepY	Step size in y direction

```
void panelStepRepeatMiddle ( Point  pStart,
                             int    iRepeatX,
                             int    iRepeatY,
                             double dStepX,
                             double dStepY
                           )
```

Step Repeat Panel Flash point = Middle

Parameters:

pStart	Position of first flashPoint
iRepeatX	Number of repeats in x direction
iRepeatY	Number of repeats in y direction
dStepX	Step size in x direction

dStepY Step size in y direction

int panelStepRepeatValidate ( )

Checks, whether blocked jobs are valid for use with panel iterator Some problems are solved automatically

Returns:

0 if it is no stepRepeat job; 1: problems were found and solved; 2: if problems were found, but not (all) solved

void pasteFromClipboard ( )

Paste all objects from clipboard onto all active layers

String peGetInputFile ( )

Gets the file used as input for the server mode.

Returns:

The file used as input for the server mode.

boolean peGetInputJobBooleanProperty (String *name*)

Gets the value as a boolean of the property with the given name for the current Panel Editor input job.

Parameters:

name * The name of the property

Returns:

The corresponding value or true if the property is not defined.

double peGetInputJobDoubleProperty (String *name*)

Gets the value as a double of the property with the given name for the current Panel Editor input job.

Parameters:

name * The name of the property

Returns:

The corresponding value or 0.0 if the property is not defined or the current job is not set.

int peGetInputJobIntegerProperty (String *name*)

Gets the value as an int of the property with the given name for the current Panel Editor input job.

Parameters:

name * The name of the property

Returns:
The corresponding value.

String peGetInputJobProperty (String *name*)

Gets the value as a String of the property with the given name for the current Panel Editor input job.

Parameters:
name * The name of the property

Returns:
The corresponding value or null if the property is not defined.

String peGetJobList ( )

Gets the value as a boolean of the property with the given name for the current Panel Editor input job.

Returns:
The corresponding value or true if the property is not defined.

int peGetOptionalQuantity ( )

Gets the optional quantity for the current PanelEditor input job.

Returns:
the optional quantity for the current PanelEditor input job.

boolean peGetPanelJobBooleanProperty (String *name*)

Gets the value as a boolean of the property with the given name for the current Panel Editor PanelJob.

Parameters:
name * The name of the property

Returns:
The corresponding value or true if the property is not defined.

double peGetPanelJobDoubleProperty (String *name*)

Gets the value as a double of the property with the given name for the current Panel Editor PanelJob.

Parameters:
name * The name of the property

Returns:
The corresponding value or 0.0 if the property is not defined or the current job is not set.

int peGetPanelJobIntegerProperty (String *name*)

Gets the value as an int of the property with the given name for the current Panel Editor PanelJob.

Parameters:

name * The name of the property

Returns:

The corresponding value.

String peGetPanelJobProperty (String *name*)

Gets the value as a String of the property with the given name for the current Panel Editor PanelJob.

Parameters:

name * The name of the property

Returns:

The corresponding value or null if the property is not defined.

int peGetPCBQuantity ( )

Gets the number of PCBs of PanelEditor input job on the current Panel job.

Returns:

the number of PCBs of PanelEditor input job on the current Panel job.

boolean peGetSingleOptimization ( )

Gets the optimization method for the server mode.

Returns:

True when single panels are generated.False when combinations are generated.

boolean peGetSolutionBooleanProperty (String *name*)

Gets the value as a boolean of the property with the given name for the current Panel Editor solution.

Parameters:

name * The name of the property

Returns:

The corresponding value or true if the property is not defined.

double peGetSolutionDoubleProperty (String *name*)

Gets the value as a double of the property with the given name for the current Panel Editor solution.

Parameters:

name * The name of the property

Returns:

The corresponding value or 0.0 if the property is not defined or the current job is not set.

int peGetSolutionIntegerProperty (String *name*)

Gets the value as an int of the property with the given name for the current Panel Editor solution.

Parameters:

name * The name of the property

Returns:

The corresponding value.

String peGetSolutionProperty (String *name*)

Gets the value as a String of the property with the given name for the current Panel Editor solution.

Parameters:

name * The name of the property

Returns:

The corresponding value or null if the property is not defined.

boolean peGetUseFrameSet ( )

Gets the useframeset method for the server mode.

Returns:

True when all frames from the set are used. False when only one frame is used.

void peSetInputFile (String *fileName*)

Sets the optional quantity for the current PanelEditor input job.

Parameters:

fileName

**void peSetInputJobProperty (String *name,*
boolean *value*
)**

Sets the value of the property with the given name to the given value for the current Input Job.

Parameters:

name * The name of the property

value * The value to assign to the property

void peSetInputJobProperty (String *name,*

double *value*
)

Sets the value of the property with the given name to the given value for the current Input Job.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetInputJobProperty (String *name,*
int *value*
)

Sets the value of the property with the given name to the given value for the current Input Job.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetInputJobProperty (String *name,*
String *value*
)

Sets the value of the property with the given name to the given value for the current Input Job.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetOptionalQuantity (int *quantity*)

Sets the optional quantity for the current PanelEditor input job.

Parameters:

quantity The value to set.

void peSetPanelJobProperty (String *name,*
boolean *value*
)

Sets the value of the property with the given name to the given value for the current Panel Editor PanelJob.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetPanelJobProperty (String *name,*

double *value*
)

Sets the value of the property with the given name to the given value for the current Panel Editor PanelJob.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetPanelJobProperty (String *name,*
int *value*
)

Sets the value of the property with the given name to the given value for the current Panel Editor PanelJob.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetPanelJobProperty (String *name,*
String *value*
)

Sets the value of the property with the given name to the given value for the current Panel Editor PanelJob.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetSingleOptimization (boolean *set*)

Sets the optimization method for the server mode.

Parameters:

set * When true, single panels are generated. When false, combinations are generated.

void peSetSolutionProperty (String *name,*
boolean *value*
)

Sets the value of the property with the given name to the given value for the current Panel Editor solution.

Parameters:

name * The name of the property
value * The value to assign to the property

void peSetSolutionProperty (String *name,*

```
                                double  value
                              )
```

Sets the value of the property with the given name to the given value for the current Panel Editor solution.

Parameters:

name * The name of the property
value * The value to assign to the property

```
void peSetSolutionProperty ( String  name,
                             int     value
                           )
```

Sets the value of the property with the given name to the given value for the current Panel Editor solution.

Parameters:

name * The name of the property
value * The value to assign to the property

```
void peSetSolutionProperty ( String  name,
                             String  value
                           )
```

Sets the value of the property with the given name to the given value for the current Panel Editor solution.

Parameters:

name * The name of the property
value * The value to assign to the property

```
void peSetUseFrameSet ( boolean  set  )
```

Sets the frame set mode method for the server mode.

Parameters:

set * When true, all frames from the set are used. When false, only one frame is used.

```
void pickAperture ( double  pt_x,
                    double  pt_y,
                    double  radius
                  )
```

Aperture Manager: Pick an Aperture near the given point

Parameters:

pt_x (X coordinate) The clicked point
pt_y (Y coordinate) The clicked point
radius Search radius around given point

**void pickAperture (Point *pt,*
double *radius*
)**

Aperture Manager: Pick an Aperture near the given point

Parameters:

pt The clicked point
radius Search radius around given point

Point pickPoint (String *sLabel*)

Wait for user pick point. It needs Ucam GUI. The function pauses script execution and waits for user interaction.

Parameters:

sLabel Label in information dialog

Returns:

Point picked by user

Exceptions:

AbortException after user abort

**void plotAddLayerToMerge (String *sJobName,*
String *sPath*
)**

Add layer to merge - to plot

Parameters:

sJobName name of job
sPath path to a dpf (layer) file, include ".dpf"

Exceptions:

AbortException

void plotAddLayerToMerge ( )

Add layer in plane 1 to merge - to plot

**boolean plotLayer (String *sPath,*
int *iFillPercentage,*
boolean *bSeparator,*
boolean *bClean*
)**

Sent layer from the path to RIP

Parameters:

sPath	path to dpf (layer) file, include .dpf
iFillPercentage	is a percentage between 0% and 100%.
bSeparator	value true/false
bClean	If true, removes the succesfully merged MergeJob instances.

Returns:

true if all is ok, false if was a problem

Exceptions:

AbortException

boolean plotLayer (int *iFillPercentage,*
boolean *bSeparator,*
boolean *bClean*
)

Sent layer in plane 1 to RIP

Parameters:

iFillPercentage	is a percentage between 0% and 100%.
bSeparator	value true/false
bClean	If true, removes the succesfully merged MergeJob instances.

Returns:

true if all is ok, false if was a problem

Exceptions:

AbortException

boolean plotMergedLayers (int *iFillPercentage,*
boolean *bSeparator,*
boolean *bClean*
)

Send all prepare layers to RIP

Parameters:

iFillPercentage	is a percentage between 0% and 100%.
bSeparator	value true/false
bClean	If true, removes the succesfully merged MergeJob instances.

Returns:

true if all is ok, false if was a problem

Exceptions:

AbortException

void plotResetParams ( )

Set plot parameters to default

void plotSetAttribute (ObjectList *oLayID,*
String *sName,*
String *sValue*
)

Set plot layer attribute

Parameters:

oLayID layer identification
sName name of attribute
sValue value of attribute

void plotSetAttribute (String *sName,*
String *sValue*
)

Set plot layer attribute

Parameters:

sName name of attribute
sValue value of attribute

Exceptions:

AbortException

void plotSetAttribute (String *sName*)

Set plot layer attribute

Parameters:

sName name of attribute

Exceptions:

AbortException

void plotSetParam (ObjectList *oLayID,*
String *sKey,*
String *sName,*
double *dValue*
)

Set plot parameter to layer with the index

Parameters:

oLayID layer identification
sKey key of parameter
sName name of parameter
dValue value of parameter

```
void plotSetParam ( String   sKey,
                    String   sName,
                    double   dValue
                  )
```

Set plot parameter

Parameters:

sKey key of parameter
sName name of parameter
dValue value of parameter

Exceptions:

AbortException

```
void plotSetParam ( ObjectList  oLayID,
                    String      sKey,
                    boolean     bValue
                  )
```

Set plot parameter to layer with the index

Parameters:

oLayID layer identification
sKey key name of parameter
bValue value of parameter

```
void plotSetParam ( String   sKey,
                    boolean  bValue
                  )
```

Set plot parameter

Parameters:

sKey key name of parameter
bValue value of parameter

Exceptions:

AbortException

```
void plotSetParam ( ObjectList  oLayID,
                    String      sKey,
                    int         iValue
                  )
```

Set plot parameter to layer with the index

Parameters:

oLayID layer identification

sKey key name of parameter
iValue value of parameter

void plotSetParam (String *sKey*,
int *iValue*
)

Set plot parameter

Parameters:
sKey key name of parameter
iValue value of parameter

Exceptions:
AbortException

void plotSetParam (ObjectList *oLayID*,
String *sKey*,
double *dValue*
)

Set plot parameter to layer with the index

Parameters:
oLayID layer identification
sKey key name of parameter
dValue value of parameter

void plotSetParam (String *sKey*,
double *dValue*
)

Set plot parameter

Parameters:
sKey key name of parameter
dValue value of parameter

Exceptions:
AbortException

void plotSetParam (ObjectList *oLayID*,
String *sKey*,
String *sValue*
)

Set plot parameter to layer with the index

Parameters:

oLayID layer identification
sKey key name of parameter
sValue value of parameter

void plotSetParam (String *sKey*,
String *sValue*
)

Set plot parameter

Parameters:
sKey key name of parameter
sValue value of parameter

Exceptions:
AbortException

void plotSetRipHost (String *sRIP*)

set Rip Host name

Parameters:
sRIP - Rip Host name

void plotStartNew ( )

Initialization Set plot parameters to default values

Point Point (Point *point*)

Create copy of a point

Parameters:
point original point

Returns:
the point

Point Point (double *x*,
double *y*,
String *units*
)

Create point from two double cordinates

Parameters:
x coordinate
y coordinate

units Ucam units

Returns:
the point

Point Point (double *x*,
double *y*
)

Create point from two double cordinates

Parameters:
x coordinate
y coordinate

Returns:
the point

void point1 (double *point1_x*,
double *point1_y*
)

Sets the Point1 Used in Numbers Dialog

Parameters:
point1_x (X coordinate) the Point1
point1_y (Y coordinate) the Point1

void point1 (Point *point1*)

Sets the Point1 Used in Numbers Dialog

Parameters:
point1 the Point1

Point point1 ( )

Gets the Point1 Used in Numbers Dialog

Returns:
Point1

void point1Active (boolean *bActivate*)

Sets the Point1 activity Used in Numbers Dialog

Parameters:
bActivate true if the Point1 should be active

boolean point1Active ( )

Returns true if the Point1 is active Used in Numbers Dialog

Returns:
true if the Point1 is active

void point1X (double *pt1X*)

Sets the Point1 x coordinate Used in Numbers Dialog

Parameters:
pt1X the Point1 x coordinate

void point1Y (double *pt1Y*)

Sets the Point1 y coordinate Used in Numbers Dialog

Parameters:
pt1Y the Point1 y coordinate

void point2 (double *point2_x*,
double *point2_y*
)

Sets the Point2 Used in Numbers Dialog

Parameters:
point2_x (X coordinate) the Point2
point2_y (Y coordinate) the Point2

void point2 (Point *point2*)

Sets the Point2 Used in Numbers Dialog

Parameters:
point2 the Point2

Point point2 ( )

Gets the Point2 Used in Numbers Dialog

Returns:
Point2

void point2Active (boolean *bActivate*)

Sets the Point2 activity Used in Numbers Dialog

Parameters:

bActivate true if the Point2 should be active

boolean point2Active ( )

Returns true if the Point2 is active Used in Numbers Dialog

Returns:

true if the Point2 is active

void point2X (double *pt2X*)

Sets the Point2 x coordinate Used in Numbers Dialog

Parameters:

pt2X the Point2 x coordinate

void point2Y (double *pt2Y*)

Sets the Point2 y coordinate Used in Numbers Dialog

Parameters:

pt2Y the Point2 y coordinate

void printListRefPoints (boolean *bOnAllActiveLay*)

Print list od reference points

Parameters:

bOnAllActiveLay if true it work on all active layers other way only on active loaded layer in plane 1

void printListRefPoints ( )

Print list od reference points on active loaded layer in plane 1

**boolean promptBoolean (String *optName,*
boolean *def*
)**

Show a checkbox field in prompt dialog

Parameters:

optName String name, will be visible in prompt dialog as a label

def boolean, default value **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

boolean a default value

double promptDouble (String *doubleName,*
double *def*
)

Show a double type field in prompt dialog

Parameters:

doubleName String name, will be visible in prompt dialog as a label
def double, default value **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

double a default value

void promptEnd ( )

Closes a prompt sequence and shows prompt dialog.

String promptFileName (String *strLabel,*
String *def*
)

Show a text field with browse button in prompt dialog

Parameters:

strLabel variable name, will be visible in prompt dialog as a label
def String, default value for file Name **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

fileName a default value

int promptInteger (String *intName,*
int *def*
)

Show an integer type field in prompt dialog

Parameters:

intName String name, will be visible in prompt dialog as a label
def int, default value **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

integer a default value

void promptLabel (String *labelText*)

Show label with given message

Parameters:

labelText String label text **Note:** if the parameter is variable, it must be defined globally for script.

Line promptLine (String *lineName,*
double *fromX,*
double *fromY,*
double *toX,*
double *toY*
)

Show four unit fields to define **Line** from **Point** X and Y and to **Point** X and Y coordinates

Parameters:

lineName String name, will be visible in prompt dialog as a label
fromX default X coordinate of the from **Point**
fromY default Y coordinate of the from **Point**
toX default X coordinate of the to **Point**
toY default Y coordinate of the to **Point** **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

Line object

Line promptLine (String *lineName,*
Line *defLine*
)

Show four unit fields to define **Line** from **Point** X and Y and to **Point** X and Y coordinates

Parameters:

lineName String name, will be visible in prompt dialog as a label
defLine default **Line** **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

Line object

String promptOption (String *optName,*
ObjectList *options,*
String *def*
)

Show a combobox field in prompt dialog

Parameters:

optName String name, will be visible in prompt dialog as a label

options ObjectList with items shown in combobox
def one of the string defined on options, default (selected) value **Note:** if the parameter is variable, it must be defined globally for script.

Returns:
String a default text

Point promptPoint (String *pointName,*
double *ptX,*
double *ptY*
)

Show two unit field to define **Point** coordinate X and Y

Parameters:
pointName String name, will be visible in prompt dialog as a label
ptX default X coordinate of the **Point**
ptY default Y coordinate of the **Point** **Note:** if the parameter is variable, it must be defined globally for script.

Returns:
Point object

Point promptPoint (String *pointName,*
Point *point*
)

Show two unit field to define **Point** coordinate X and Y

Parameters:
pointName String name, will be visible in prompt dialog as a label
point default **Point** **Note:** if the parameter is variable, it must be defined globally for script.

Returns:
Point object

Rectangle promptRectangle (String *rectangleName,*
double *xmin,*
double *xmax,*
double *ymin,*
double *ymax*
)

Show four unit fields to define **Rectangle** X min, X max, Y min and Y max coordinates

Parameters:
rectangleName String name, will be visible in prompt dialog as a label
xmin default minimal rectangle X coordinate
xmax default maximal rectangle X coordinate
ymin default minimal rectangle Y coordinate
ymax default maximal rectangle Y coordinate **Note:** if the parameter is variable, it must be

defined globally for script.

Returns:
Rectangle object

**Rectangle promptRectangle (String *rectangleName,*
Rectangle *rectangle*
)**

Show four unit fields to define **Rectangle** X min, X max, Y min and Y max coordinates

Parameters:

rectangleName	String name, will be visible in prompt dialog as a label
rectangle	default **Rectangle** **Note:** if the parameter is variable, it must be defined globally for script.

Returns:
Rectangle object

**void promptStart (String *sSetName,*
String *sTitle*
)**

Initiates prompt dialog for users input and sets the name of the variables set Must be the first command before a prompt commands sequence

Parameters:

sSetName	the name of the variable set. If it is `null` or is empty it is ignored.
sTitle	the PromptDialog title. If it is `null` or is empty it is ignored **Note:** if the parameter is variable, it must be defined globally for script.

---

void promptStart (String *sSetName*)

Initiates prompt dialog for users input and sets the name of the variables set Must be the first command before a prompt commands sequence

Parameters:

sSetName	the name of the variable set. If it is `null` or is empty it has the same meaning as **promptStart()**. **Note:** if the parameter is variable, it must be defined globally for script.

---

void promptStart ( )

Initiates prompt dialog for users input Must be the first command before a prompt commands sequence

---

**String promptString (String *strName,*
String *def*
)**

Show a text field in prompt dialog

Parameters:

strName String name, will be visible in prompt dialog as a label
def String, default value **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

String a default value

double promptUnit (String *unitName,*
double *def,*
String *units*
)

Show an unit field in prompt dialog

Parameters:

unitName String name, will be visible in prompt dialog as a label
def double, default value
units "mm", "mil" or "inch" to determine units of the default value **Note:** if the parameter is variable, it must be defined globally for script.

Returns:

a double value in current units

void qmerge (String *sOptions*)

qmerge description

Parameters:

sOptions qmerge options

void quitBlockEdit (boolean *bSave,*
boolean *bKeepLink*
)

Aperture Manager: Quits the Block Definition Edit Mode

Parameters:

bSave true: save changes; false: discard changes
bKeepLink true: keeps block links

void quitBlockEdit (boolean *bSave*)

Aperture Manager: Quits the Block Definition Edit Mode. The saved (if bSave is `true`) block keeps link when Ucam.db key `uiape_block_keeplink` is set to 1

Parameters:

bSave true: save changes; false: discard changes

**void quitBlockMultiEdit (boolean *bSave,*
boolean *bKeepLink*
)**

Aperture Manager: Quits the Block Definition Multi Edit Mode

Parameters:

bSave true: save changes; false: discard changes
bKeepLink true: keeps block links

void quitBlockMultiEdit (boolean *bSave*)

Aperture Manager: Quits the Block Definition Multi Edit Mode. The saved (if bSave is `true`) block keeps link when Ucam.db key `uiape_block_keeplink` is set to 1

See also:

quitBlockEdit(boolean, boolean)

Parameters:

bSave true: save changes; false: discard changes

void quitComplexEdit (boolean *bSave*)

Aperture Manager: Quits the Complex Definition Edit Mode

Parameters:

bSave true: save changes; false: discard changes

void quitConfirm ( )

Quit application after asking for confirmation.

void readAmli (String *sPath*)

read AMLI files

Parameters:

sPath - path to the root directory

Exceptions:

IOException

void recognizeContours (String *sConName*)

recognizeContours

Parameters:

sConName The name of the conversion file/contour.

Rectangle Rectangle (Rectangle *rect*)

Create copy of a rectangle

Parameters:

rect original rectangle

Returns:

the rectangle

Rectangle Rectangle (double *xmin*,
double *ymin*,
double *xmax*,
double *ymax*,
String *units*
)

Create rectangle from four coordinates

Parameters:

xmin left boundary of rectangle
ymin bottom boundary of rectangle
xmax right boundary of rectangle
ymax top boundary of rectangle
units Ucam units

Returns:

the rectangle

Rectangle Rectangle (double *xmin*,
double *ymin*,
double *xmax*,
double *ymax*
)

Create rectangle from four coordinates

Parameters:

xmin left boundary of rectangle
ymin bottom boundary of rectangle
xmax right boundary of rectangle
ymax top boundary of rectangle

Returns:

the rectangle

Rectangle Rectangle (Point *ptPoint1*,

Point *ptPoint2*
)

Create enclosing rectangle of two points

Parameters:

ptPoint1 first point
ptPoint2 second point

Returns:

the rectangle

void redo ( )

Redo an action

void registerJobOnPoints ( )

Does a 3 point transformation on a job/layer to correct distortion of scanned artwork. In the layer(s) point 1 to 6 must be defined and the registration places point 1 on 4, 2 on 5 and 3 on 6. All objects are moved with the average x and y movement of those points.

void registerLayers ( )

Performs automatic layer registration on the job. All active layers are registered on a reference layer. The layer in plane 3 is taken as a reference layer The selected pads in the reference layer are taken as a reference for registration.

void registerOnGrid (double *GridStep_x*,
double *GridStep_y*,
double *GridOri_x*,
double *GridOri_y*,
double *dXRadius*,
double *dYRadius*
)

Snaps all (selected) objects to the grid specified. Works on selected objects.

Parameters:

GridStep_x (X coordinate) The grid step
GridStep_y (Y coordinate) The grid step
GridOri_x (X coordinate) The grid origin
GridOri_y (Y coordinate) The grid origin
dXRadius Only objects within the specified distance in x direction are moved.
dYRadius Only objects within the specified distance in y direction are moved.

void registerOnGrid (Point *GridStep*,

```
                    Point   GridOri,
                    double  dXRadius,
                    double  dYRadius
                  )
```

Snaps all (selected) objects to the grid specified. Works on selected objects.

Parameters:

GridStep	The grid step
GridOri	The grid origin
dXRadius	Only objects within the specified distance in x direction are moved.
dYRadius	Only objects within the specified distance in y direction are moved.

```
void registerOnPads ( double   dRadius,
                      boolean  bOnFlashPoint
                    )
```

Registers a job/layer on a reference layer. All pads that are not registered and all tracks are offset with the average offset of the pads. The layer in plane 2 is taken as a reference layer. Works on selected objects.

Parameters:

dRadius	Maximum distance between pads. If the distance between a pad and a reference pad is bigger than the radius, the pad is not registered.
bOnFlashPoint	If true register only on pad flashpoint (not the edges)

```
void removeApeAttr (  )
```

removes all aperture attributes on layer lay

```
void removeJobAttr (  )
```

removes all job attributes

```
void removeLayAttr (  )
```

removes all layer attributes on layer lay

```
void removeNetAttr ( int     iNetNumber,
                     String  sAttrName,
                     String  sAttrValue
                   )
```

Removes all net attributes with given name on job and given net

Parameters:

iNetNumber	Net number
sAttrName	Net attribute name

sAttrValue Net attribute value

void removeNetAttr (int *iNetNumber,*
String *sAttrName*
)

Removes all net attributes with given name on job and given net

Parameters:

iNetNumber Net number
sAttrName Net attribute name

void removeNetAttr (String *sAttrName,*
String *sNetName*
)

Removes all net attributes with given name and value on job

Parameters:

sAttrName Net attribute name
sNetName

void removeNetAttr (String *sAttrName*)

Removes all net attributes with given name on job

Parameters:

sAttrName Net attribute name

void removeObjAttr ( )

removes all object attributes on layer lay

void removeObjectAttribute (String *sAttrName,*
String *sAttrValue*
)

removeObjectAttribute Sets the object attribute with the given name and value. Takes objects from active layers from current job by Ucam options. (e.g. all selected objects)

Parameters:

sAttrName The object attribute name
sAttrValue The object attribute value

void removeObjectAttribute (String *sAttrName*)

removeObjectAttribute remove the object attribute with the given name. Takes objects from active layers from current job by Ucam options. (e.g. all selected objects)

Parameters:

sAttrName The object attribute name

void removeYsphotechPlotstamp (int *plotstampID*)

removes the plotstamp with a certain id

Parameters:

plotstampID the ID of the plotstamp to be removed *

void replaceApeByCurrent ( )

Replace selected objects on all active layers by current aperture

void replaceApertures ( )

Aperture Manager: Replace apertures of selection on current layer by current aperture

void replaceBitmapByContours ( )

replaceBitmapByContours

void replaceDrawsWithArcs (double *tolerance*)

Tries to replace chains of minimum 3 consecutive vectors by an arc within given tolerance. If the operation succeeds, all the vectors are deleted and a new arc is created. The start/end points of the vector chain become the start and end points of the new arc.

Parameters:

tolerance The tolerance is the maximum distance from the arc to any of the vectors being replaced.

void replaceInnersByOuters ( )

Replace inner countours by outer contours

**void replaceZeroLengthDraws (double *dMaxLength,*
boolean *bFunctional,*
boolean *bNonFunctional*
)**

Replace Zero Length Draws and Arcs with flashes

Parameters:

dMaxLength	Maximum length of objects to be replaced
bFunctional	Replace objects within functional copper if true
bNonFunctional	Replace objects within non-functional copper if true

void reproducePanel (String *report*)

Panel reproduce

Parameters:

report	Report file (.prf)

void reset ( )

Reset all values in Numbers

void resetCFMEE ( )

Resets CFMEE settings: removes all alignment points

void resetCores (int *iTop*,
String *sAttach*
)

Reset Cores between given Layers

Parameters:

iTop	top Layer index
sAttach	"top", "bottom", "none" or "both"

void resetCores ( )

Reset All Cores

void resetWorkspace ( )

resets current workspace layout.

void resetYsphotech ( )

Resets Ysphotech settings: removes all alignment points and plotstamps

void restoreArcs (boolean *bPreferFullArc*)

Restore Arcs Repair Invalid Arcs

Parameters:

bPreferFullArc The parameter influence image for arcs where it does not clear if they are full arcs or zero arcs. If the value is set to true they will be full arcs, other way they will be zero arcs.

void restoreContours (boolean *bPreferFullArc*)

Restore Contours Repair Contours and Complexes

Parameters:

bPreferFullArc The parameter influence image for arcs where it does not clear if they are full arcs or zero arcs. If the value is set to true they will be full arcs, other way they will be zero arcs.

void returnVariables (ObjectList *returnVariables*)

Terminates called script and returns ObjectList of variables

Parameters:

returnVariables ObjectList of the return variables

Exceptions:

AbortException after user abort

See also:

runScriptWithReturn(String)

void reverse ( )

Reverse selected elements

void reverseLayer ( )

Reverse the active layers

void reverseLayers ( )

Reverse active layer(s)

void rotate (double *angle*,
boolean *bUseCenter*,
boolean *bOnRefPoints*

)

Rotate selections around origin corresponds to: Transform Objects - Edit - Rot 90/Rot270/Rot angle

Parameters:

angle	Value of the angle (pos:ccw or neg:cw)
bUseCenter	If true, rotate is done around center
bOnRefPoints	If true, rotate is also applied to reference points

void roundDraw (double *pt_x,*
double *pt_y,*
double *dis*
)

Replace a draw by a rounded join

Parameters:

pt_x	(X coordinate) **Point** on draw
pt_y	(Y coordinate) **Point** on draw
dis	Radius of rounding (see fillet) enter 0 to make radius = length of draw / 2

void roundDraw (Point *pt,*
double *dis*
)

Replace a draw by a rounded join

Parameters:

pt	**Point** on draw
dis	Radius of rounding (see fillet) enter 0 to make radius = length of draw / 2

void routStatistics (String *doOption*)

Create the rout report for the current job.

Parameters:

doOption	Either "all" or "sel".

void routStatistics ( )

Create the rout report for the current job.

void runDRC (String *sCfgFile,*
boolean *bBuildNetlist,*
String *sUseNetlist,*
boolean *bSelErrors*

)

Run Smart DRC check

Parameters:

sCfgFile	Configuration file
bBuildNetlist	Build Netlist (true, false)
sUseNetlist	Use Netlist ("none", "layer" or "job")
bSelErrors	Select errors

String runFile (String *sScriptPath,*
ObjectList *argv*
)

Run script from file(s). Uses variables according to `argv` parameter. if `argv` is `null` the call is the same as **`runFile(String sScript)`** and uses all variables defined in parent interpreter if exists if `argv` is `null` the call is the same as **`runFile(String sScript)`** and uses all variables defined in parent interpreter if exists if `argv` is empty `[{}]` the script is called completelly without input variable if `argv` is eg. `[{"name", 1, true}]` the called script knows one ObjectList variable with the name "args". The items are accessable by the index. Eg. `name = args[0];` and `isAvailable = args[2];`

Parameters:

sScriptPath	file name which is either full path, wildcard mask or just base name, in this case the file is searched in current directory or using `hyperscript.script.path` ucam.db key.
argv	ObjectList with input arguments [{arg1, arg2, ..., argn}] or `null` or empty ObjectList [{}]

Returns:

String RUN_STATUS_OK if run is OK, otherwise returns message about issue encountered.

Exceptions:

AbortException if any problem encountered during script call

See also:

runFile(String)

runFile(String)

String runFile (String *sScriptPath*)

Run script from file(s)

Parameters:

sScriptPath	file name which is either full path, wildcard mask or just base name, in this case the file is searched in current directory or using `hyperscript.script.path` ucam.db key.

Returns:

String RUN_STATUS_OK if run is OK, otherwise returns message about issue encountered.

Exceptions:

AbortException if any problem encountered during script call

String runScript (String *sScript,*
ObjectList *argv*
)

Run (interprete) given text. Uses variables according to `argv` parameter. if `argv` is `null` the call is the same as **`runScript(String sScript)`** and uses all variables defined in parent interpreter if exists if `argv` is `null` the call is the same as **`runScript(String sScript)`** and uses all variables defined in parent interpreter if exists if `argv` is empty `[{}]` the script is called completelly without input variable if `argv` is eg. `[{"name", 1, true}]` the called script knows one ObjectList variable with the name "args". The items are accessable by the index. Eg. `name = args[0];` and `isAvailable = args[2];`

Parameters:

sScript text of the script.

argv ObjectList with input arguments [{arg1, arg2, ..., argn}] or `null` or empty ObjectList [{}]

Returns:

String RUN_STATUS_OK if run is OK, otherwise returns message about issue encountered.

Exceptions:

AbortException if any problem encountered during script call

See also:

runScript(String)

runScript(String)

String runScript (String *sScript*)

Run (interprete) given text **Example:**

```
promptStart();
    script = promptString("String to execute", "saveJob();");
promptEnd;
runScript(script);
runScript(script);
```

Parameters:

sScript text of the script.

Returns:

String RUN_STATUS_OK if run is OK, otherwise returns message about issue encountered.

Exceptions:

AbortException if any problem encountered during script call

ObjectList runScriptWithReturn (String *sScript,*
Object[] *argv*
)

Run (interprete) given text. Uses variables according to `argv` parameter. if `argv` is `null` the call is the same as **`runScriptWithReturn(String sScript)`** and uses all variables defined in parent interpreter if exists if `argv` is `null` the call is the same as **`runScriptWithReturn(String sScript)`** and uses all variables defined in parent interpreter if exists if `argv` is empty `[{}]` the script is called completelly without input variable if `argv` is eg. `[{"name", 1, true}]` the called script knows one ObjectList variable with the name "args". The items are accessable by the index. Eg. `name = args[0];` and `isAvailable = args[2];`

Parameters:

sScript text of the script.

argv ObjectList with input arguments [{arg1, arg2, ..., argn}] or `null` or empty ObjectList [{}]

Returns:

An ObjectList with returning values.

Exceptions:

AbortException if any problem encountered during script call

ObjectList runScriptWithReturn (String *sScript*)

Run (interprete) given text. Uses all variables defined in parent interpreter if exists.

Parameters:

sScript text of the script.

Returns:

An ObjectList with returning values.

Exceptions:

AbortException if any problem encountered during script call

void saveAmli (String *sPath*)

save AMLI files

Parameters:

sPath - path to the root directory

void saveBuildup (String *sSpec,*
String *sCustomer,*
String *sDrcPar,*
String *sCoreRef,*
String *sPrePregMat,*
String *sCopperMat,*
String *sJobFlow,*
String *sTechCheck,*
String *sAttrSet,*
String *sDatumList,*
ObjectList *layList*
)

Save Current job buildup

Parameters:

sSpec The file specification for the buildup file.

sCustomer the customer

sDrcPar the drc parameter file specification

sCoreRef uCoreMaterial attribute value

sPrePregMat uPrePregMaterial attribute value
sCopperMat uCopperMaterial attribute value
sJobFlow uJobFlow attribute value
sTechCheck uTechnologycheck attribute value
sAttrSet uAttributeset attribute value
sDatumList comma separated list of datums
layList Array of the layer name mapping

int saveJob ( )

Save Current job

int saveJobAs (String *fullPath,*
String *sVersion*
)

Save Current job as...

Parameters:
fullPath The full pathname for the current job The target directory must already exist
sVersion version "3" or "6" or "9"

int saveJobAs (String *fullPath*)

Save Current job as...

Parameters:
fullPath The full pathname for the current job The target directory must already exist

void saveJobAsV3 ( )

Save Current job as version 3. A separate function is needed for the button, to handle the Licensed case well.

void saveJobAsV6 ( )

Save Current job as version 6. A separate function is needed for the button, to handle the Licensed case well.

void saveJobAsV9 ( )

Save Current job as version 9. A separate function is needed for the button, to handle the Licensed case well.

void saveLayer (String *sClass,*
String *sSubClass,*

int *iLayIndex,*
String *sFullPath*
)

Save Layer As

Parameters:

sClass	Layer class "layer", "drill" or "extra"
sSubClass	Layer subclass eg. "outline", "mask", "silk", ...
iLayIndex	Layer index in given class or in given subclass
sFullPath	String new full path file name

void saveLayer (String *layName,*
String *fullPath*
)

Save Layer with name

Parameters:

layName	Name of layer to be saved
fullPath	The full pathname for the dpf layer The target directory must already exist

void saveMessagesAs (String *sFilePath*)

Saves the Messages window content with the given file path

Parameters:

sFilePath	The messages will be saved to the file with given path

void saveOrder ( )

Saves the defined order after all modifications to the rout order have been made.

void saveSplitConfig (String *sConfigName*)

Save split configuration for the given name

Parameters:

sConfigName	name of configuration

void saveUFD (String *sUFDName*)

Saves the contents of the current fault database to the file with the given specification.

Parameters:

sUFDName	UFD file specification

void saveWorkspace ( )

Save currently set workspace layout. NOTE: The same as menu command Workspaces > Save

void saveWorkspace (String *sWorkspaceName*)

Save workspace layout as file with a given name in XML format.

Parameters:

sWorkspaceName

void saveWorkspaceAs (String *sWorkspaceName*)

Save currently set workspace layout as file with a given name. NOTE: The same as menu command Workspaces > Save as...

Parameters:

sWorkspaceName

void scale (double *dScaleValue,*
boolean *bUseCenter*
)

Scale selections

Parameters:

dScaleValue Scale factor (greater than 1 for up and less than 1 for down)
bUseCenter If true, Scale is done around center

void scaleObjectOnAttribute (String *sName,*
double *dScaleFactor,*
double *dMinClearance*
)

Scales all objects (flash, draw, arc, txt) that have attribute 'name'. Scale 'factor' defines margin between objects which may define together scale zone. All objects in zone are scaled with factor. The center is used as center of the enclosing rectangle of the objects in a zone.

Parameters:

sName an attribute name.
dScaleFactor a scale factor.
dMinClearance a minimal clearance between characters that should be respected after enlarging the text.

void screendump ( )

Makes a screendump of the selected area and gives you the ability to print it.

```
void secureEtchCompensation ( double   dPadSpread,
                              double   dSmdSpread,
                              double   dTrackSpread,
                              double   dAreaSpread,
                              double   dPadPadClearance,
                              double   dPadSmdClearance,
                              double   dPadTrackClearance,
                              double   dPadAreaClearance,
                              double   dSmdSmdClearance,
                              double   dSmdTrackClearance,
                              double   dSmdAreaClearance,
                              double   dTrackTrackClearance,
                              double   dTrackAreaClearance,
                              double   dAreaAreaClearance,
                              String   sContourMethod,
                              boolean  bProcessSameNetSpacing,
                              boolean  bBackupOriginalLayer,
                              boolean  bCheckMissingCopper,
                              boolean  bFastMode,
                              int      iShiftMode,
                              double   dMinCopper
                            )
```

Secure Etch Compensation

Parameters:

dPadSpread	The spread value to apply to circular pads
dSmdSpread	The spread value to apply to non circular pads
dTrackSpread	The spread value to apply to draws
dAreaSpread	The spread value to apply to areas
dPadPadClearance	The clearance to keep between circular pads
dPadSmdClearance	The clearance to keep between circular pads and non circular pads
dPadTrackClearance	The clearance to keep between circular pads and draws
dPadAreaClearance	The clearance to keep between circular pads and areas
dSmdSmdClearance	The clearance to keep between non circular pads
dSmdTrackClearance	The clearance to keep between non circular pads and draws
dSmdAreaClearance	The clearance to keep between non circular pads and areas
dTrackTrackClearance	The clearance to keep between draws
dTrackAreaClearance	The clearance to keep between draws and areas
dAreaAreaClearance	The clearance to keep between areas
sContourMethod	The contour compensation method ("spread", "stroke", or "circle")
bProcessSameNetSpacing	Whether to process spacing between objects of the same net as well
bBackupOriginalLayer	If true, make backups of the original layer(s)
bCheckMissingCopper	If true, report errors when original copper was removed
bFastMode	If true, skip slow "select embedded" step to make SEC faster but sometimes less correct
iShiftMode	Mode to shift clearances, 0: no shift, 1: shift to cut only areas, 2: like 1, but keep original copper of areas
dMinCopper	The minimum copper width to keep

```
void secureEtchCompensation ( double  dPadSpread,
                              double  dSmdSpread,
                              double  dTrackSpread,
                              double  dAreaSpread,
                              double  dPadPadClearance,
                              double  dPadSmdClearance,
                              double  dPadTrackClearance,
                              double  dPadAreaClearance,
                              double  dSmdSmdClearance,
                              double  dSmdTrackClearance,
                              double  dSmdAreaClearance,
                              double  dTrackTrackClearance,
                              double  dTrackAreaClearance,
                              double  dAreaAreaClearance,
                              String  sContourMethod,
                              boolean bProcessSameNetSpacing,
                              boolean bBackupOriginalLayer,
                              boolean bCheckMissingCopper,
                              boolean bFastMode,
                              int     iShiftMode
                            )
```

Secure Etch Compensation

Parameters:

dPadSpread	The spread value to apply to circular pads
dSmdSpread	The spread value to apply to non circular pads
dTrackSpread	The spread value to apply to draws
dAreaSpread	The spread value to apply to areas
dPadPadClearance	The clearance to keep between circular pads
dPadSmdClearance	The clearance to keep between circular pads and non circular pads
dPadTrackClearance	The clearance to keep between circular pads and draws
dPadAreaClearance	The clearance to keep between circular pads and areas
dSmdSmdClearance	The clearance to keep between non circular pads
dSmdTrackClearance	The clearance to keep between non circular pads and draws
dSmdAreaClearance	The clearance to keep between non circular pads and areas
dTrackTrackClearance	The clearance to keep between draws
dTrackAreaClearance	The clearance to keep between draws and areas
dAreaAreaClearance	The clearance to keep between areas
sContourMethod	The contour compensation method ("spread", "stroke", or "circle")
bProcessSameNetSpacing	Whether to process spacing between objects of the same net as well
bBackupOriginalLayer	If true, make backups of the original layer(s)
bCheckMissingCopper	If true, report errors when original copper was removed
bFastMode	If true, skip slow "select embedded" step to make SEC faster but sometimes less correct
iShiftMode	Mode to shift clearances, 0: no shift, 1: shift to cut only areas, 2: like 1, but keep original copper of areas

```
void secureEtchCompensation ( double  dPadSpread,
                              double  dSmdSpread,
```

```
                    double  dTrackSpread,
                    double  dAreaSpread,
                    double  dPadPadClearance,
                    double  dPadSmdClearance,
                    double  dPadTrackClearance,
                    double  dPadAreaClearance,
                    double  dSmdSmdClearance,
                    double  dSmdTrackClearance,
                    double  dSmdAreaClearance,
                    double  dTrackTrackClearance,
                    double  dTrackAreaClearance,
                    double  dAreaAreaClearance,
                    String  sContourMethod,
                    boolean bProcessSameNetSpacing,
                    boolean bBackupOriginalLayer,
                    boolean bCheckMissingCopper,
                    boolean bFastMode
                )
```

Secure Etch Compensation

Parameters:

dPadSpread	The spread value to apply to circular pads
dSmdSpread	The spread value to apply to non circular pads
dTrackSpread	The spread value to apply to draws
dAreaSpread	The spread value to apply to areas
dPadPadClearance	The clearance to keep between circular pads
dPadSmdClearance	The clearance to keep between circular pads and non circular pads
dPadTrackClearance	The clearance to keep between circular pads and draws
dPadAreaClearance	The clearance to keep between circular pads and areas
dSmdSmdClearance	The clearance to keep between non circular pads
dSmdTrackClearance	The clearance to keep between non circular pads and draws
dSmdAreaClearance	The clearance to keep between non circular pads and areas
dTrackTrackClearance	The clearance to keep between draws
dTrackAreaClearance	The clearance to keep between draws and areas
dAreaAreaClearance	The clearance to keep between areas
sContourMethod	The contour compensation method ("spread", "stroke", or "circle")
bProcessSameNetSpacing	Whether to process spacing between objects of the same net as well
bBackupOriginalLayer	If true, make backups of the original layer(s)
bCheckMissingCopper	If true, report errors when original copper was removed
bFastMode	If true, skip slow "select embedded" step to make SEC faster but sometimes less correct

void secureEtchCompensationUndo ( )

Secure Etch Compensation Undo Undoes any previous SEC. Deletes the negative clearance apertures and shrinks any objects carrying the spread value in the attribute named by ucam.db key "ClearanceManager.spreadAttribute".

void selectAll ( )

Select all objects.

void selectAll (String *selectMode*)

Select or deselect all objects.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectAllApertures ( )

Aperture Manager: Select all Objects of all Apertures in Aperture list

void selectAllContours (String *selectMode*,
String *conMode*
)

Select all contours.

Parameters:

selectMode	Either + (select) or - (deselect)
conMode	Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

void selectAllContours (String *selectMode*,
double *xSize*,
double *ySize*,
String *conMode*
)

Select or deselect all contours.

Parameters:

selectMode	Either + (select) or - (deselect)
xSize	The maximum width (x size) for the contour enclosing box.
ySize	The maximum height (y size) for the contour enclosing box.
conMode	Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

void selectAmbiguousContours (String *selectMode*)

Select ambiguous contours.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectAperture (ObjectList *apeIndexArray*)

Aperture Manager: Select Objects of Apertures

Parameters:

apeIndexArray Array of indexes of the apertures on the current layer

void selectAperture ( )

Aperture Manager: Select Objects of current Aperture

**void selectAperturesBiggerThan (String *selectMode,*
double *dx,*
double *dy*
)**

Select apertures bigger than ...

Parameters:

selectMode Either + (select) or - (deselect)
dx X value of enclosing rectangle
dy Y value of enclosing rectangle

**void selectAperturesSmallerThan (String *selectMode,*
double *dx,*
double *dy*
)**

Select apertures smaller or equal than ...

Parameters:

selectMode Either + (select) or - (deselect)
dx X value of enclosing rectangle
dy Y value of enclosing rectangle

**void selectByApeAttributeNames (String *selectMode,*
String[] *sName*
)**

Select or deselect all objects of the specified attribute names.

Parameters:

selectMode Either + (select) or - (deselect)
sName

void selectByApertureShape (String *selectMode,*

String *apertureShapes*
)

Deprecated:
Select or deselect all objects of the specified object type.

Parameters:

selectMode	Either + (select) or - (deselect)
apertureShapes	Comma separated list of cir, don, rec, squ, box, oct, com, the, con, tex, blo

void selectByAttributeName (String *selectMode,*
String *sName*
)

Deprecated:
Select or deselect all objects of the specified attribute name.

Parameters:

selectMode	Either + (select) or - (deselect)
sName	- attribute name

void selectByAttributeValue (String *selectMode,*
String *sName,*
String *sValue*
)

Deprecated:
Select or deselect all objects of the specified attribute value.

Parameters:

selectMode	Either + (select) or - (deselect)
sName	- attribute name
sValue	- attribute value

void selectByObjectType (String *selectMode,*
String *objectTypes*
)

Deprecated:
Select or deselect all objects of the specified object type.

Parameters:

selectMode	Either + (select) or - (deselect)
objectTypes	Comma separated list of f (flash), d (draw), a (arc) or v (vector text)

void selectChained (String *selectMode,*
double *pt_x,*
double *pt_y*
)

Select chained draws.

Parameters:

selectMode	Either + (select) or - (deselect)
pt_x	(X coordinate) start point
pt_y	(Y coordinate) start point

void selectChained (String *selectMode,*
Point *pt*
)

Select chained draws.

Parameters:

selectMode	Either + (select) or - (deselect)
pt	start point

void selectChained (String *selectMode,*
double *pt_x,*
double *pt_y,*
double *dTolerance*
)

Select chained draws.

Parameters:

selectMode	Either + (select) or - (deselect)
pt_x	(X coordinate) start point
pt_y	(Y coordinate) start point
dTolerance	radius of click

void selectChained (String *selectMode,*
Point *pt,*
double *dTolerance*
)

Select chained draws.

Parameters:

selectMode	Either + (select) or - (deselect)
pt	start point
dTolerance	radius of click

void selectChainedObjects (String *selectMode,*
double *pnt_x,*
double *pnt_y,*
double *pixelRadius,*

```
                    double  dOffCenter,
                    boolean bSameApe,
                    boolean bSameOrientation
                )
```

Parameters:

selectMode	Either "+" or "-" to define type of selection
pnt_x	(X coordinate) clicked point
pnt_y	(Y coordinate) clicked point
pixelRadius	radius around pnt, tolerance to find an object
dOffCenter	Tolerance between end point of reference object and start point of the following object
bSameApe	If true the chain may only constist of objects of the same aperture
bSameOrientation	If true the object must have the same orientation as the reference

```
void selectChainedObjects ( String  selectMode,
                            Point   pnt,
                            double  pixelRadius,
                            double  dOffCenter,
                            boolean bSameApe,
                            boolean bSameOrientation
                          )
```

Parameters:

selectMode	Either "+" or "-" to define type of selection
pnt	clicked point
pixelRadius	radius around pnt, tolerance to find an object
dOffCenter	Tolerance between end point of reference object and start point of the following object
bSameApe	If true the chain may only constist of objects of the same aperture
bSameOrientation	If true the object must have the same orientation as the reference

```
void selectContourByClick ( String  selectMode,
                            double  pt_x,
                            double  pt_y,
                            String  conMode
                          )
```

Select contour by click (De)selects a region in a job/layer. A region is a copper area defined by 1 outer contour and 1 or more inner contours. The region should enclose target point.

Parameters:

selectMode	Either + (select) or - (deselect)
pt_x	(X coordinate) The target point.
pt_y	(Y coordinate) The target point.
conMode	Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

```
void selectContourByClick ( String  selectMode,
```

```
                    Point   pt,
                    String  conMode
                )
```

Select contour by click (De)selects a region in a job/layer. A region is a copper area defined by 1 outer contour and 1 or more inner contours. The region should enclose target point.

Parameters:

selectMode	Either + (select) or - (deselect)
pt	The target point.
conMode	Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

```
void selectContoursInWindow ( String  selectMode,
                              double  rect_xmin,
                              double  rect_ymin,
                              double  rect_xmax,
                              double  rect_ymax,
                              String  conMode
                            )
```

Select contours inside selection window (De)selects all regions in a job/layer inside a rectangular area. A region is a copper area defined by 1 outer contour and 1 or more inner contours.

Parameters:

selectMode	Either "+" (select) or "-" (deselect)
rect_xmin	(left boundary of rectangle) The target rectangle.
rect_ymin	(bottom boundary of rectangle) The target rectangle.
rect_xmax	(right boundary of rectangle) The target rectangle.
rect_ymax	(top boundary of rectangle) The target rectangle.
conMode	Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

```
void selectContoursInWindow ( String     selectMode,
                              Rectangle  rect,
                              String     conMode
                            )
```

Select contours inside selection window (De)selects all regions in a job/layer inside a rectangular area. A region is a copper area defined by 1 outer contour and 1 or more inner contours.

Parameters:

selectMode	Either "+" (select) or "-" (deselect)
rect	The target rectangle.
conMode	Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

```
void selectContoursWithThinRegion ( double  dThin,
                                    String  selectMode
                                  )
```

Select contours with thin region.

Parameters:

dThin	value in current units
selectMode	Either + (select) or - (deselect)

void selectCurrentAperture (String *selectMode*)

Select all objects using current aperture.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectCurrentApertureDefinition (String *selectMode*)

Select all objects using current aperture definition and the current aperture number.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectCurrentObject (String *selectMode*)

Select current object.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectDoubles (String *selectMode,*
double *tolerance*
)

Select doubles. (De)selects all objects that are duplicated.

Parameters:

selectMode	Either "+" or "-". Selects when "+", deselects when "-".
tolerance	The tolerance on the aperture definition.

void selectEmbedded (String *selectMode,*
double *tolerance*
)

Select embedded. (De)selects all embedded objects.

Parameters:

selectMode	Either "+" or "-". Selects when "+", deselects when "-".
tolerance	Tolerance on aperture definition

**void selectFlashesLongerThan (String *selectMode,*
double *rRefRatio*
)**

Select/deselect flashes in the job which the long side is over dRefRatio times longer than the short side

Parameters:

selectMode - the option "+" (select) or "-" (deselect)
rRefRatio - the referent ratio

**int selectHornablePads (String *selectMode,*
String *apertureShapes*
)**

Add hornable pads (rec and/or box) to selection or remove them from selection. Shows an error to the user if some parameters where invalid

Parameters:

selectMode Either "+" (select) or "-" (deselect), default value is "+"
apertureShapes options are "rec" or "box" default value is "rec,box".

Returns:

Number of (de)selected hornable pads

boolean selectInvalidArcs ( )

Select Invalid Arcs

Returns:

false

**int selectInvalidArcs (String *sSelectMode,*
double *dDeviation,*
String *sLimit*
)**

Select/Deselect invalid arcs by parameters

Parameters:

sSelectMode "+" for select invalid arcs or "-" deselect deselect arcs
dDeviation is difference from distance Center point to To point or Center point to From point
sLimit ">" for select invalid arcs with the deviation bigger than the value dDeviation "<" for select invalid arcs with the deviation smaller than the value dDeviation

Returns:

number of selected invalid arcs

void selectIsolatedFlashes (String *selectMode*)

Select/deselect isolated flashes

Parameters:

selectMode Either + (select) or - (deselect) objects

void selectMesh (String *selectMode*)

Select mesh.

Parameters:

selectMode Either + (select) or - (deselect)

int selectNetByClick (String *selectMode,*
double *pt_x,*
double *pt_y*
)

Select net by click on layer in plane 1 then 2 and then 3.

Parameters:

selectMode Either + (select) or - (deselect) objects
pt_x (X coordinate) clicked location
pt_y (Y coordinate) clicked location

Returns:

clicked net number at the point

int selectNetByClick (String *selectMode,*
Point *pt*
)

Select net by click on layer in plane 1 then 2 and then 3.

Parameters:

selectMode Either + (select) or - (deselect) objects
pt clicked location

Returns:

clicked net number at the point

int selectNetByName (String *selectMode,*
String *sNetName*
)

Select net by name in current job (active layers).

Parameters:

selectMode Either "+" (select) or "-" (deselect) objects
sNetName Net name

void selectNetByNumber (String *selectMode,*
int *net,*
boolean *bSelectShaved,*
boolean *bSelectBroken*
)

Select net by number in current job (active layers).

Parameters:

selectMode	Either + (select) or - (deselect) objects
net	Net number
bSelectShaved	Specifies if shaved objects should be selected, or not.
bSelectBroken	Specifies if broken objects should be selected, or not.

void selectNetByNumber (String *selectMode,*
int *net*
)

Select net by number in current job (active layers).

Parameters:

selectMode	Either + (select) or - (deselect) objects
net	Net number

void selectNetByTestpoints (String *selectMode,*
int *nbt*
)

Select net by number of testpoints.

Parameters:

selectMode	Either + (select) or - (deselect) objects
nbt	number of testpoints

void selectNetsWithoutPads (String *selectMode*)

Select nets without pads.

Parameters:

selectMode	Either + (select) or - (deselect) objects

String selectNonFunctionalPads ( )

selectNonFunctionalPads Selects the non functional pads in the current job.

**void selectObjectAttribute (String *sAttrName,*
String *sAttrValue*
)**

Deprecated:
selectObjectAttribute Select objects with set attribute with the given name and value from current job.

Parameters:
sAttrName The object attribute name
sAttrValue The object attribute value

void selectObjectAttribute (String *sAttrName*)

Deprecated:
selectObjectAttribute Select objects with set attribute with the given name from current job.

Parameters:
sAttrName The object attribute name

**void selectObjectByAttribute (String *sAttrName,*
String *sAttrValue*
)**

selectObjectAttribute Select objects with set attribute with the given name and value from current job.

Parameters:
sAttrName The object attribute name
sAttrValue The object attribute value

void selectObjectByAttribute (String *sAttrName*)

selectObjectAttribute Select objects with set attribute with the given name from current job.

Parameters:
sAttrName The object attribute name

**void selectObjectByAttributeName (String *selectMode,*
String *sName*
)**

Select or deselect all objects of the specified attribute name.

Parameters:
selectMode Either + (select) or - (deselect)
sName - attribute name

void selectObjectByAttributeValue (String *selectMode,*

String *sName*,
String *sValue*
)

Select or deselect all objects of the specified attribute value.

Parameters:

selectMode	Either + (select) or - (deselect)
sName	- attribute name
sValue	- attribute value

void selectObjectByShape (String *selectMode*,
String *apertureShapes*
)

Select or deselect all objects of the specified object type.

Parameters:

selectMode	Either + (select) or - (deselect)
apertureShapes	Comma separated list of cir, don, rec, squ, box, oct, com, the, con, tex, blo

void selectObjectByType (String *selectMode*,
String *objectTypes*
)

Select or deselect all objects of the specified object type.

Parameters:

selectMode	Either + (select) or - (deselect)
objectTypes	Comma separated list of f (flash), d (draw), a (arc) or v (vector text)

void selectOpenContours (String *selectMode*)

Select open contours.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectOverlappingContours (String *selectMode*)

Select overlapping contours.

Parameters:

selectMode	Either + (select) or - (deselect)

void selectOverlaps (String *selectMode*)

Select overlapping objects.

Parameters:

selectMode Either + (select) or - (deselect)

void selectPainted (String *selectMode*)

Select painted objects. Painted objects are a collection of tracks that touch and/or overlap.

Parameters:

selectMode Either "+" or "-". Selects when "+", deselects when "-".

int selectPaintedAreas (boolean *bUseLoops,*
boolean *bExcludeChains*
)

Selects all painted data in a current layer. Painted data are a collection of tracks that are covered on one side by other tracks.

Parameters:

bUseLoops Use a loop with a number of iterations

bExcludeChains Exclude chains from the painted area

int selectPaintedAreas ( )

Selects all painted data in a current layer. Painted data are a collection of tracks that are covered on one side by other tracks.

void selectPlotStamps (String *selectMode*)

Select or deselect all plot stamps.

Parameters:

selectMode Either + (select) or - (deselect)

void selectPolygon (boolean *bInside,*
boolean *bOutside,*
boolean *bCrossing,*
String *sShapes,*
String *sObjects,*
String *selectMode,*
ObjectList *polygonPoints*
)

Select Polygon.

Parameters:

bInside if true, select objects inside the polygon
bOutside if true, select objects outside the polygon
bCrossing if true, select object crossing the polygon outline
sShapes Possible values are "cir", "rec", "squ", "box", "oct", "con", "com", "the", "blo" and "txt", "don", "und". Multiple shapes can be specified in a comma separated list. Specify null, or an empty string to select all shapes.

Example: "cir,box,com".

Parameters:

sObjects The object name to (de)select. Possible values are "f", "d", "a" and "v" for flash, draw, arc and vtext. Multiple objects can be specified. Specify null, or an empty string to select all objects.

Example: "da", or "d,a".

Parameters:

selectMode Either + (select) or - (deselect) objects
polygonPoints outline of selections

void selectReferenceLayer (String *selectMode*)

Select objects with flashpoint inside reference layer.

Parameters:

selectMode Either + (select) or - (deselect) objects

void selectReverse (String *selectMode*)

Select objects with reverse polarity.

Parameters:

selectMode Either + (select) or - (deselect) objects

void selectSmallContours (String *selectMode*,
double *xSize*,
double *ySize*,
String *conMode*
)

Select small contours. (De)selects contours which size is smaller than the given dimensions.

Parameters:

selectMode Either "+" or "-". Selects when "+", deselects when "-".
xSize The maximum width (x size) for the contour enclosing box.
ySize The maximum height (y size) for the contour enclosing box.
conMode Defines what should be selected. "i" for inner contours, "o" for outer contours and "r" for regions.

void selectSmallSurface (String *selectMode*,
double *surface*,

```
                    String  conMode
                )
```

Select small contour surfaces.

Parameters:

selectMode	Either + (select) or - (deselect)
surface	Surface in square unit
conMode	Region, Outer or Inner

```
void selectSmallTracks ( String  selectMode,
                         String  lenMode,
                         String  dstMode,
                         double  maxLength
                       )
```

Select small tracks.

Parameters:

selectMode	Either + (select) or - (deselect)
lenMode	"abs" for length or "rel" for ratio
dstMode	"ctc" for center to center, "oto" for outside to outside
maxLength	Maximum length

```
void selectTouchingObjects ( String  selectMode,
                             double  pnt_x,
                             double  pnt_y,
                             double  pixelRadius
                           )
```

Parameters:

selectMode	Either "+" or "-" to define type of selection
pnt_x	(X coordinate) clicked point
pnt_y	(Y coordinate) clicked point
pixelRadius	radius around pnt, tolerance to find an object

```
void selectTouchingObjects ( String  selectMode,
                             Point   pnt,
                             double  pixelRadius
                           )
```

Parameters:

selectMode	Either "+" or "-" to define type of selection
pnt	clicked point
pixelRadius	radius around pnt, tolerance to find an object

```
void selectWindow ( String  selectMode,
```

```
                    double  rect_xmin,
                    double  rect_ymin,
                    double  rect_xmax,
                    double  rect_ymax,
                    String  winopt,
                    String  sShapes,
                    String  sObjects
                )
```

Select window using board coordinates (De)selects all data in a rectangular area.

Parameters:

selectMode	Either "+" or "-" to select, or deselect.
rect_xmin	(left boundary of rectangle) rectangular select area
rect_ymin	(bottom boundary of rectangle) rectangular select area
rect_xmax	(right boundary of rectangle) rectangular select area
rect_ymax	(top boundary of rectangle) rectangular select area
winopt	Add "i", "o", and/or "c" to select inside, outside, and/or crossing.

Example: "ic" select all object in rectangle and all crossing objects. objects.

Parameters:

sShapes	The shape name to (de)select. Possible values are "cir", "rec", "squ", "box", "oct", "con", "com", "the", "blo", "txt", "don" and "und". Multiple shapes can be specified in a comma separated list. Specify null, or an empty string to select all shapes. Example: "cir,box,com".
sObjects	The object name to (de)select. Possible values are "f", "d", "a" and "v" for flash, draw, arc and vtext. Multiple objects can be specified. Specify null, or an empty string to select all objects.

Example: "da", or "d,a".

```
void selectWindow ( String      selectMode,
                    Rectangle   rect,
                    String      winopt,
                    String      sShapes,
                    String      sObjects
                )
```

Select window using board coordinates (De)selects all data in a rectangular area.

Parameters:

selectMode	Either "+" or "-" to select, or deselect.
rect	rectangular select area
winopt	Add "i", "o", and/or "c" to select inside, outside, and/or crossing.

Example: "ic" select all object in rectangle and all crossing objects. objects.

Parameters:

sShapes	The shape name to (de)select. Possible values are "cir", "rec", "squ", "box", "oct", "con", "com", "the", "blo", "txt", "don" and "und". Multiple shapes can be specified in a comma separated list. Specify null, or an empty string to select all shapes. Example: "cir,box,com".
sObjects	The object name to (de)select. Possible values are "f", "d", "a" and "v" for flash, draw, arc and vtext. Multiple objects can be specified. Specify null, or an empty string to select all objects.

Example: "da", or "d,a".

**void selectZeroLengthDraws (double *dMaxLength,*
boolean *bFunctional,*
boolean *bNonFunctional*
)**

Select Zero Length Draws and Arcs

Parameters:

dMaxLength	Maximum length of objects to be selected
bFunctional	Select objects within functional copper if true
bNonFunctional	Select objects within non-functional copper if true

boolean setApe (int *index*)

Sets current aperture

Parameters:

index - index of the aperture in the aperture list. Starts by 1 and index must be smaller or equal to aperture count in the aperture list. If the index is incorrect the current aperture is set to `null`

Returns:

`true` if the index is correct and current aperture is set, `false` if the index is incorrect and current aperture is NOT set.

**void setApertureAttribute (String *sAttributeName,*
String *sAttributeValue*
)**

Aperture Manager: Add or Modify an Attribute of current Aperture

Parameters:

sAttributeName	The new name of the Attribute
sAttributeValue	The new value of the Attribute

**void setAttributeOnObject (String *attrName,*
String *attrValue*
)**

Deprecated:

This function is GUI ONLY, replaced by addObjectAttribute(sAttrName, sAttrValue)

See also:

addObjectAttribute(String sAttrName, String sAttrValue) Insert attribute on objects

Parameters:

attrName	Name of attribute
attrValue	Value of attribute

boolean setCurrentAperture (int *iIndex*)

Sets current aperture

Parameters:

iIndex Index of the aperture in the aperture list. Starts by 1 and index must be smaller or equal to aperture count in the aperture list. If the index is incorrect the current aperture is set to `null`

Returns:

`true` if the index is correct and current aperture is set, `false` if the index is incorrect and current aperture is NOT set.

**void setInPlane (int *newPlane,*
int *iIndex*
)**

Set layer in plane color.

Parameters:

newPlane Target plane color
iIndex Layer index in job build-up

**void setInPlane (int *newPlane,*
String *layName*
)**

Set layer in plane color.

Parameters:

newPlane Target plane color
layName Layer name

**void setInPlane (int *newPlane,*
String *layClass,*
String *laySubclass,*
String *attach,*
int *index*
)**

Set layer in plane color and activate/deactivate according to plane action setup in function **VHS.setInPlane**(...). setup in function **VHS.setInPlane**(...).

Parameters:

newPlane Target plane color
layClass Layer class
laySubclass Layer subclass
attach Attachment top or bottom (only for extra layers)
index Index within the specified class or subclass e.g for a 6 layer job To set the bottom copper layer to plane 1:

```
setInPlane(1, "layer", null, "", 6);
```


```
setInPlane(1, "layer", null, "", 6);
```


The value 6 refers to the 6th layer or

```
setInPlane(1, "layer", "outer", "", 2);
```


```
setInPlane(1, "layer", "outer", "", 2);
```


The value 2 refers to the second layer of subclass "outer". For layers of subclass "extra" the attach mode is also taken into account. To set a bottom mask with index 2 to plane 1:

```
setInPlane(1, "extra", "mask", "bottom", 2);
```


```
setInPlane(1, "extra", "mask", "bottom", 2);
```


Any extra layer can be set to any plane by using its index within the extra layer class. All "extra" layers attached to "top" are numbered from top to bottom, for "extra" layers attached to "bottom" the numbering continues from bottom to top. To set the extra layer with index 4 to plane 1:

```
setInPlane(1, "extra", null, "", 4);
```


```
setInPlane(1, "extra", null, "", 4);
```


Index within the specified drill class or subclass Drill layers are numbered from left to right To set the first plated drill layer of a job with 1 unplated and 2 plated layers to plane 1:

```
setInPlane(1, "drill", null , "", 2);
```


```
setInPlane(1, "drill", null , "", 2);
```


The value 2 refers to the second drill layer or

```
setInPlane (1, "drill", "plated", "", 1);
```


```
setInPlane (1, "drill", "plated", "", 1);
```


The value 1 refers to the first plated drill layer. **Note:** The `extra.order` ucam.db key can change the order of the extra layers. Scripts relying on the index of extra layers could fail to execute correctly on other systems.

void setInPlane (int *newPlane,*
String *layClass,*
String *laySubclass,*
String *attach,*
int *index,*
boolean *activate*
)

Set layer in plane color and activate/deactivate.

Parameters:

newPlane	Target plane color
layClass	Layer class
laySubclass	Layer subclass
attach	Attachment top or bottom (only for extra layers)
index	Index within the specified class or subclass e.g for a 6 layer job To set the bottom copper layer to plane 1:

```
setInPlane(1, "layer", null, "", 6, true);
```


```
setInPlane(1, "layer", null, "", 6, true);
```


The value 6 refers to the 6th layer or

```
setInPlane(1, "layer", "outer", "", 2, true);
```


```
setInPlane(1, "layer", "outer", "", 2, true);
```


The value 2 refers to the second layer of subclass "outer" For layers of subclass "extra" the attach mode is also taken into account. To set a bottom mask with index 2 to plane 2 and deactivate:

```
setInPlane(2, "extra", "mask", "bottom", 2, false);
```


```
setInPlane(2, "extra", "mask", "bottom", 2, false);
```


Any extra layer can be set to any plane by using its index within the extra layer class. All "extra" layers attached to "top" are numbered from top to bottom, for "extra" layers attached to "bottom" the numbering continues from bottom to top. To set the extra layer with index 4 to plane 2 and activate:

```
setInPlane(2, "extra", null, "", 4, true);
```


```
setInPlane(2, "extra", null, "", 4, true);
```


Index within the specified drill class or subclass Drill layers are numbered from left to right To set the first plated drill layer of a job with 1 unplated and 2 plated layers to plane 3 and activate:

```
setInPlane(3, "drill", null, "", 2, true);
```


```
setInPlane(3, "drill", null, "", 2, true);
```


The value 2 refers to the second drill layer or

```
setInPlane (3, "drill", "plated", "", 1, true);
```


```
setInPlane (3, "drill", "plated", "", 1, true);
```


The value 1 refers to the first plated drill layer **Note:** The `extra.order` ucam.db key can change the order of the extra layers. Scripts relying on the index of extra layers could fail to execute correctly on other systems.

activate

activate Activation of target plane

void setInPlane (int *newPlane,*
ObjectList *layerID,*
boolean *activate*
)

Set layer in plane color and activate/deactivate.

Parameters:

newPlane Target plane color
layerID the layer ID given e.g. by `getLayerID()` function
activate Activation of target plane

See also:

HSH_base::getLayerID(Ulayer)

HSH_base::getLayerID(Ulayer, String)

void setInPlaneByName (int *newPlane,*
String *layName,*
boolean *activate*
)

Set layer in plane color and activate/deactivate.

Parameters:

newPlane Target plane color
layName Layer name
activate Activation of target plane

void setLayerViewBottom (ObjectList *nameArray*)

Sets layer names to view in the dialog Job View in the part Bottom View

Parameters:

nameArray array of layer names

void setLayerViewDrill (ObjectList *nameArray*)

Sets layer names to view in the dialog Job View in the part Drill View

Parameters:

nameArray array of layer names

void setLayerViewMain (ObjectList *nameArray*)

Sets layer names to view in the dialog Job View in the part Main View

Parameters:

nameArray array of layer names

void setLayerViewTop (ObjectList *nameArray*)

Sets layer names to view in the dialog Job View in the part Top View

Parameters:

nameArray array of layer names

void setMode (ObjectList *oMode*)

Sets the current operation mode, units and snapmode

Parameters:

oMode - array of [sOption, sUnit, sSnapMode]

See also:

setMode(String, String, String)

setMode(String, String, String)

void setMode (String *sParams*)

Sets the current operation mode, units and snapmode parameter can contain all value combinations, they should be separated by comma or space

Parameters:

sParams - ("sel" or "all" or "selall") and/or ("mm" or "mil" or "inch") and/or ("no", "closest", "midpoint",

"intersection", "closestline", "virtualintersection", "layerintersection", "grid")

void setMode (String *sOption,*
String *sUnit,*
String *sSnapMode*
)

Sets the current operation mode, units and snapmode

Parameters:

sOption - "sel" or "all" or "selall"
sUnit - "mm" or "mil" or "inch"
sSnapMode - snapmode : "no", "closest", "midpoint", "intersection", "closestline", "virtualintersection", "layerintersection", "grid"

Ulayer setNextLayerToPlane1 ( )

Change layer in plane 1 by default rules (class and index).

Returns:

new layer in plane 1

void setOrigin (double *p_x,*
double *p_y*
)

Set Origin using parameters

Parameters:

p_x (X coordinate) point to set the origin
p_y (Y coordinate) point to set the origin

void setOrigin (Point *p*)

Set Origin using parameters

Parameters:

p point to set the origin

void setOrigin (double *pt_x,*
double *pt_y,*
boolean *bOnRefPoints*
)

Set job origin for active layers using board coordinates

Parameters:

pt_x (X coordinate) New DPF zero, relative to current DPF zero
pt_y (Y coordinate) New DPF zero, relative to current DPF zero
bOnRefPoints If true, move is also applied to reference points

void setOrigin (Point *pt,*
boolean *bOnRefPoints*
)

Set job origin for active layers using board coordinates

Parameters:
pt New DPF zero, relative to current DPF zero
bOnRefPoints If true, move is also applied to reference points

void setOriginCenter (double *p_x,*
double *p_y,*
boolean *useOutline*
)

Set Origin Center using parameters

Parameters:
p_x (X coordinate) point to set the origin center
p_y (Y coordinate) point to set the origin center
useOutline use outline

void setOriginCenter (Point *p,*
boolean *useOutline*
)

Set Origin Center using parameters

Parameters:
p point to set the origin center
useOutline use outline

void setOriginToCenter (boolean *bUseOutline,*
boolean *bOnRefPoints*
)

Set job origin for active layers to center of job

Parameters:
bUseOutline If true, center of outline layer is used
bOnRefPoints If true, move is also applied to reference points

void setPlotParam (String *sKey,*
int *iValue*
)

Sets the plot parameter.

Parameters:

sKey The key. Possible values are :

- "RESOLUTION"
- "XOFF" (the x offset)
- "YOFF" (the y offset)
- "XSCALE" (the x scale)
- "YSCALE" (the y scale)
- "SXCEN" (the x scale center)
- "SYCEN" (the y scale center)
- "MXCEN" (the x mirror center)
- "MYCEN" (the y mirror center)
- "POSITION":
- - MergeJob.PLOT_POSITION_COMBINE_FILL_PERCENTAGE
- - MergeJob.PLOT_POSITION_COMBINE
- - MergeJob.PLOT_POSITION_SINGLE
- - MergeJob.PLOT_POSITION_SINGLE_LEFT_FIXED
- - MergeJob.PLOT_POSITION_SINGLE_RIGHT_FIXED
- "ENLARGE_VECTORS_MINIMUMSIZE"
- "ENLARGE_VECTORS_AMOUNT"
- "ENLARGE_FLASH_MINIMUMSIZE"
- "ENLARGE_FLASH_AMOUNT"
- "ENLARGE_CONDUCTOR_SIZE" (Adds a size. Use 0 to remove them all.)
- "ENLARGE_CONDUCTOR_AMOUNT"
- "ENLARGE_COMPLEX"
- "APPLY_ENLARGE_TO" (The value must be one of Ulayer.PLOT_ENLARGE_*)
- "VARIABLE_TEXT_HEIGHT"
- "VARIABLE_TEXT_DIRECTION" (The value must be one of Ulayer.PLOT_VARTEXT_DIRECTION_*)

iValue The integer value.

void setPlotParam (String *sKey,*
double *dValue*
)

Sets the plot parameter.

Parameters:

sKey The key. Possible values are :

- "XOFF" (the x offset)
- "YOFF" (the y offset)
- "XSCALE" (the x scale)
- "YSCALE" (the y scale)
- "SXCEN" (the x scale center)
- "SYCEN" (the y scale center)
- "MXCEN" (the x mirror center)
- "MYCEN" (the y mirror center)
- "ENLARGE_VECTORS_MINIMUMSIZE"
- "ENLARGE_VECTORS_AMOUNT"
- "ENLARGE_FLASH_MINIMUMSIZE"

"ENLARGE_FLASH_AMOUNT"
- "ENLARGE_CONDUCTOR_SIZE" (adds a size, or removes them all if size equals 0)
- "ENLARGE_CONDUCTOR_AMOUNT"
- "ENLARGE_COMPLEX"
- "VARIABLE_TEXT_HEIGHT"

dValue The double value.

**void setPlotParam (String *sKey,*
String *sValue*
)**

Sets the plot parameter.

Parameters:

sKey The key. Possible values are :

- "POLARITY" ("POSITIVE" or "NEGATIVE")
- "EMULSION" ("DOWN" or "UP")
- "MIR" ("N", "X", "Y" or "XY")
- "ROT" ("Y", "N", "F")
- "FILM"
- "VARIABLE_TEXT"

sValue The String value.

void setResolution (int *resolution*)

Set resolution used by Ucam

Parameters:

resolution - resolution in number of internal units per mil

void setSnap (String *sMode*)

This method can only set a single snap mode.

Parameters:

sMode - snapmode : "no", "closest", "midpoint", "intersection", "closestline", "virtualintersection", "layerintersection", "grid"

void setSnapOnContour (boolean *on*)

turn on/off the snapping on contour

Parameters:

on true - turn on the snapping on contour, false - turn off the snapping on contour

void setUnit (String *unit*)

Set Unit used by Ucam

Parameters:

unit - "mm" or "mil" or "inch"

void setYsphotechAlignmentPointType (int *region*,
int *point*,
String *type*)

set the type of the alignment point

Parameters:

region	the regionnumber of the point (0 for global)
point	the alignmentpoint number (1 to 4)
type	the type of the alignment point (can be "matrix" or "default")

void setYsphotechPlotstamp (int *plotstampID*,
double *rec_xmin*,
double *rec_ymin*,
double *rec_xmax*,
double *rec_ymax*,
String *type*)

adds or changes the plotstamp with a certain id

Parameters:

plotstampID	the ID of the plotstamp. (should be unique)
rec_xmin	(left boundary of rectangle) the enclosing rectangle of the plotstamp
rec_ymin	(bottom boundary of rectangle) the enclosing rectangle of the plotstamp
rec_xmax	(right boundary of rectangle) the enclosing rectangle of the plotstamp
rec_ymax	(top boundary of rectangle) the enclosing rectangle of the plotstamp
type	the type of the plotstamp

void setYsphotechPlotstamp (int *plotstampID*,
Rectangle *rec*,
String *type*)

adds or changes the plotstamp with a certain id

Parameters:

plotstampID	the ID of the plotstamp. (should be unique)
rec	the enclosing rectangle of the plotstamp
type	the type of the plotstamp

void shavePads (double *dPadTraClr,*
double *dPadPadClr,*
int *iClip,*
boolean *bShaveInsideCom*
)

Shave Pads

Parameters:

dPadTraClr - the pad to track clearance value. If the value is negative or zero then the clearance between pad and track is ignored.

dPadPadClr - the pad to pad clearance value. If the value is negative or zero then the clearance between pads is ignored.

iClip - If 1, objects are clipped. If 0, reverse objects are inserted.

bShaveInsideCom true - shave COM regions between each other, false - shave only with foreign objects.

void shavePads (double *dPadTraClr,*
double *dPadPadClr,*
int *iClip*
)

Shave Pads

Parameters:

dPadTraClr - the pad to track clearance value. If the value is negative or zero then the clearance between pad and track is ignored.

dPadPadClr - the pad to pad clearance value. If the value is negative or zero then the clearance between pads is ignored.

iClip - If 1, objects are clipped. If 0, reverse objects are inserted.

void shavePadsOnMaskLayer (double *dPadToTrack,*
double *dPadToPad*
)

Shave Soldermask layer

Parameters:

dPadToTrack - Enter the pad to track clearance value. If the value is negative or zero then the clearance between pad and track is ignored.

dPadToPad - Enter the pad to pad clearance value. If the value is negative or zero then the clearance between pads is ignored.

void showBlockStructure ( )

show Block Structure Information dialog

void showMeasureValues (double *p1_x,*
double *p1_y,*

```
                        double dx,
                        double dy,
                        double clr,
                        double rng
                    )
```

Shows measure values in Numbers dialog if it is available otherwise the values are stored for future use.

Parameters:

p1_x	(X coordinate) Point1
p1_y	(Y coordinate) Point1
dx	the offset of the Point2 in X
dy	the offset of the Point2 in X
clr	clearance value
rng	ring value

```
void showMeasureValues ( Point  p1,
                        double dx,
                        double dy,
                        double clr,
                        double rng
                    )
```

Shows measure values in Numbers dialog if it is available otherwise the values are stored for future use.

Parameters:

p1	Point1
dx	the offset of the Point2 in X
dy	the offset of the Point2 in X
clr	clearance value
rng	ring value

```
void showNetlistProfile (  )
```

Displays the netlist profile in the terminal window.

```
void silkOptimize ( int     iReference,
                    double  dClearanceToReference,
                    boolean bCompensateBumps,
                    double  dBumpClearance,
                    int     iMethod,
                    double  dMinimumDrawLength
                  )
```

Clips all active layers of the current job using the given parameters.

Parameters:

iReference	Defines the reference data used to perform the clipping against. Either REFERENCE_PLANE2, REFERENCE_COPPER,

	REFERENCE_COPPER_PADS, REFERENCE_MASK, REFERENCE_COPPER_FREE_OF_MASK or REFERENCE_COPPER_PADS_FREE_OF_MASK
dClearanceToReference	The clearance value to be added to the reference before clipping.
bCompensateBumps	When true, the bumpClearance value is used to clip all data that touches copper which is covered by mask. Only applies when the reference is REFERENCE_COPPER_FREE_OF_MASK or REFERENCE_COPPER_PADS_FREE_OF_MASK
dBumpClearance	The clearance value to be added to clip data the touches copper which is covered by mask. Only applies when compensateBumps is true and bumpClearance is bigger than clearanceToReference.
iMethod	Defines the clipping method. Either METHOD_EXACT, METHOD_REVERSE or METHOD_SPLIT_DRAWS.
dMinimumDrawLength	The minimum length of draws that can remain after splitting draws. Only applies when the METHOD_SPLIT_DRAWS is specified.

int smoothen (String *mode,*
double *dMaxDeviation*
)

See also:

smoothen(Ulayer, String, double, int, boolean, int)

smoothen(Ulayer, String, double, int, boolean, int)

Parameters:

mode
dMaxDeviation

int smoothen (String *mode,*
double *dMaxDeviation,*
int *iMinReplacePoints*
)

Smoothens contours in the currently active layer by approximating groups of draws and arcs by larger arcs or longer draws.

Parameters:

mode	Determines wheter we can use arcs, draws or both to smoothen a contour. Can be "arc" or "both", "line" is planned.
dMaxDeviation	Maximal allowed deviation between the original draw and its replacement
iMinReplacePoints	The minimal amount of points to consider for an approximation Default is 4 (which is equivalent to three tracks in default mode)

Returns:

amount of contours that where smoothened or an error code if something went wrong

void spawn_func (String *sCommand*)

Spawn function in separate thread

Parameters:

sCommand - the command to spawn

int splitContour (double *dOverlapX,*
double *dOverlapY,*
double *dMinX,*
double *dMinY*
)

Function split contour. Input is current aperture from active layer in plane 1. Output is contour or contours without inners. Overlap has to be smaller or equal than Min. Also clean all smaller inners according MinX or MinY.

Parameters:

dOverlapX distance overlaping between new results contoures in X coordinate, iOverlapX =< iMinX
dOverlapY distance overlaping between new results contoures in Y coordinate, iOverlapY =< iMinY
dMinX all chains must be bigger than this
dMinY all chains must be bigger than this

Returns:

Count of the split contours or 0 if no contours have to be split (no inners) or -1 if function failed

void splitContours ( )

Split countours

boolean stackupByGerAttr ( )

change order of layers by Gerber X2 layer attributes

Returns:

true if OK, false if fault was detected

void standardizeBoxes ( )

Standardize boxes on all active layers

void testVDPathfinder (int *iStart,*
int *iEnd*
)

Test default VDPathfinder between node iStart and iEnd

Parameters:

iStart
iEnd

void testVDPathfinder2 (double *dStartX*,
double *dStartY*,
double *dEndX*,
double *dEndY*
)

Test default VDPathfinder between coordinates given

Parameters:
dStartX
dStartY
dEndX
dEndY

void toggleApertureSelections ( )

Aperture Manager: Toggle Selections in current layer

void toggleSelections ( )

Toggle selections.

void toggleViewInBlocks ( )

Toggle Show In Blocks

void toggleViewMode ( )

Toggle View Mode Filled->Skeleton->Outline->Filled...

void toggleViewObjects ( )

Toggle Show Objects

void toggleViewRefPoints ( )

Toggle Show Reference Points

void toggleViewZero ( )

Toggle Show DPF Zero

**void trimDraws (double *p1_x,*
double *p1_y,*
double *p2_x,*
double *p2_y*
)**

Trim (connect) 2 virtually intersecting draws

Parameters:

p1_x (X coordinate) **Point** on draw 1
p1_y (Y coordinate) **Point** on draw 1
p2_x (X coordinate) **Point** on draw 2
p2_y (Y coordinate) **Point** on draw 2

**void trimDraws (Point *p1,*
Point *p2*
)**

Trim (connect) 2 virtually intersecting draws

Parameters:

p1 **Point** on draw 1
p2 **Point** on draw 2

void undo ( )

Undo an action

void undoClear ( )

Clear undo history

**void unload (String *sClass,*
String *sSubClass,*
int *iLayIndex,*
boolean *bSave*
)**

Unload Layer

Parameters:

sClass	Layer class "layer", "drill" or "extra"
sSubClass	Layer subclass eg. "outline", "mask", "silk", ...
iLayIndex	Layer index in given class or in given subclass
bSave	true if the Layer should be saved before unloading

void updateProbes ( )

Update Probes

void updateTestPoints ( )

Update Test points

void updateTestPointsAndProbes ( )

Update Test points and probe information

void updateZPosition ( )

Recalculates the buildup Z-positions.

void utestCheck3DProbeClearance (boolean *bCheck3DProbeClearance*,
double *dClearance*
)

Sets parameter to dialog Utest

Parameters:

bCheck3DProbeClearance	check clearance
dClearance	value of clearance

void utestCreateEtmComponentLayers (boolean *bCreateCompLay*)

Sets parameter to dialog Utest for Create Component Layers

Parameters:

bCreateCompLay	sets to create the etm component layers

void utestDedicatedFixtures (boolean *bDedicatedFixtures*,
boolean *bFirstProbeNumber0*,
double *dFontScale*,
int *iShowProbeNumberEvery*,
boolean *bShowConnectorNumberAlways*,
boolean *bContinousNumbering*,
boolean *bContinousNumberingOnBottom*
)

Sets parameters to dialog Utest and to subdialog Dedicated Fixtures

Parameters:

bDedicatedFixtures	enable dialog box to display the Dedicated Fixture-screen
bFirstProbeNumber0	Indicate whether numbering should start with 0
dFontScale	Specify the fontsize you want to use for the probe-numbers.
iShowProbeNumberEvery	Specify the interval between two probe-numbers.
bShowConnectorNumberAlways	Indicate whether or not the connector-number should be included in the probe-number.
bContinousNumbering	if set true, numbering will increment the number with one for each probe.if set false, numbering will reset the probe number for every new connector.
bContinousNumberingOnBottom	Indicate whether or not numbering should continue or restart numbering on bottom.

void utestDo ( )

Generate all data for an electrical test machine

void utestFiducals (boolean *bFiducals*)

Sets parameters to dialog Utest

Parameters:

bFiducals enable the dialog with settings

void utestFixtureSizeSplit (boolean *bFixtureSizeSplit,*
int *iSplitSet*
)

Sets parameters to dialog Utest and to subdialog Fixture Size Split

Parameters:

bFixtureSizeSplit New layers sessions probe layers are added to the top and the bottom of the job. The new layers are of class EXTRA and subclass probe.

iSplitSet

void utestGuidePlates (boolean *bGuidePlates*)

Sets parameter to dialog Utest

Parameters:

bGuidePlates sets to generate the guideplate layers.

void utestKelvin4WireTest (boolean *bKelvin4WireTest,*
boolean *bUsedMidPoints,*
boolean *bTestOnBlindHoles,*
boolean *bTestOnlyThroughHoles,*
boolean *bTestAllPads,*
double *dMinDrillDia,*

```
                        double  dMaxDrillDia,
                        int     iSearchDepthLimit
                      )
```

Sets parameters to dialog Utest and to subdialog Kelvin4WireTest

Parameters:

bKelvin4WireTest	enable settings dialog
bUsedMidPoints	Set true if will midpoints used
bTestOnBlindHoles	Set true if will blind holes used
bTestOnlyThroughHoles	Set true if will Through Holes used
bTestAllPads	test all pads connected to an existing kelvin tested hole
dMinDrillDia	min. drill diameter value
dMaxDrillDia	max. drill diameter value
iSearchDepthLimit	max. search depth value

```
void utestKelvin4WireTest ( boolean  bKelvin4WireTest,
                            boolean  bUsedMidPoints,
                            boolean  bTestOnBlindHoles,
                            boolean  bTestOnlyThroughHoles,
                            double   dMinDrillDia,
                            double   dMaxDrillDia,
                            int      iSearchDepthLimit
                          )
```

Sets parameters to dialog Utest and to subdialog Kelvin4WireTest

Parameters:

bKelvin4WireTest	enable settings dialog
bUsedMidPoints	Set true if will midpoints used
bTestOnBlindHoles	Set true if will blind holes used
bTestOnlyThroughHoles	Set true if will Through Holes used
dMinDrillDia	min. drill diameter value
dMaxDrillDia	max. drill diameter value
iSearchDepthLimit	max. search depth value

```
void utestMachine ( int     iSession,
                    String  sMachName,
                    String  sAccesstype
                  )
```

Sets values to dialog Utest to machine section

Parameters:

iSession	number of session
sMachName	name of machine
sAccesstype	access type - side

```
void utestMicroAdjustment ( boolean  bMicroAdjustment,
```

```
                    int     iNbrOfTestPoints,
                    double  dTestPointDiameter,
                    double  dTestPointShiftEdge,
                    double  dTestPointShiftValue,
                    double  dTestPointPitch,
                    double  dClearanceFactor,
                    double  dCenterDiameter
                  )
```

Sets parameters to dialog Utest and to subdialog Micro Adjustment

Parameters:

bMicroAdjustment	enable Micro Adjustment Setup
iNbrOfTestPoints	The following alignment point parameters are used for all selected test points.
dTestPointDiameter	The aperture diameter of the pads used as alignment points (in the current unit).
dTestPointShiftEdge	The distance between the test point and the first alignment point's center (in the current unit).
dTestPointShiftValue	The distance between each of the alignment points' center on the axis of the shortest side (in the current unit).
dTestPointPitch	The distance between each of the alignment points' center on the axis of the longest side (in the current unit).
dClearanceFactor	The minimum clearance required between other copper areas and the edge of the test point (at the longest side).
dCenterDiameter	The aperture diameter of the center point (in the current unit).

```
void utestNetlist ( boolean  bNetlist,
                    boolean  bNetlistBuild,
                    boolean  bNetlistExpand
                  )
```

Sets parameters to dialog Utest to section Netlist

Parameters:

bNetlist	if sets true, generate the netlist for the current job.
bNetlistBuild	if sets true, generates or regenerates a netlist of the job.
bNetlistExpand	if sets true, generates the netlist for a panelized job

```
void utestOutput ( boolean  bOutput,
                   boolean  bDrillFixture,
                   String   sDrillFixture,
                   boolean  bNetlist,
                   String   sNetlist,
                   boolean  bElectTest,
                   boolean  bPinInserter,
                   boolean  bRepairAid,
                   String   sRepairAid
                 )
```

Sets parameters to dialog Utest and subdialog Output

Parameters:

bOutput	enable dialog with settings
bDrillFixture	enable drill fixture
sDrillFixture	sets drill fixture machine
bNetlist	Generates the Netlist Output files.
sNetlist	Generates the electrical test files in IPC-D-356 format.
bElectTest	Generates the Electrical Test files.
bPinInserter	Generates the Output file for the pin inserter, currently only available for PL-2000, an ATG machine.
bRepairAid	Generates the Repair Aid file for the repair machine.
sRepairAid	sets to RAID or EPC or Circuitest

Exceptions:

AbortException

```
void utestProbeAssignment ( boolean  bProbeAssignment,
                            boolean  bStagger,
                            String   sStagpnt,
                            boolean  bStagopttrian,
                            boolean  bStagoptlined,
                            double   dPitch,
                            double   dTolerance,
                            double   dSetBack,
                            boolean  bReverse,
                            boolean  bAxis
                          )
```

Sets parameters to dialog Utest and to subdialog Probe Assignment

Parameters:

bProbeAssignment	New layers are added to the top and the bottom of the job. The new layers are of class EXTRA and subclass probe.
bStagger	Moves SMD test points in the test point layers so as to avoid probe clashes.
sStagpnt	This menu allows you to select 2-points, 3-points or 4-points staggering. Depending on your choice 2, 3 or 4 positions are used on the SMD for assigning the probe.
bStagopttrian	determine the pattern of the 3-points or 4-points stagger.
bStagoptlined	determine the pattern of the 3-points or 4-points stagger.
dPitch	Defines the maximum center to center distance between two SMD pads. If the SMD pitch between two SMD pads is smaller than the maximum pitch then these SMD pads are staggered.
dTolerance	Enter a value to define the tolerance for finding SMD rows and columns. If the distance between the SMD pads (=SMD pitch) varies more than the tolerance value, the SMD pads no longer belong to the same row or column (for staggering).
dSetBack	Moves the staggered probes over a defined distance back towards the flash point of the SMD. This causes a border of the defined distance at the SMD edge where no probes are assigned. The Setback value is the distance between the test pin and the edge of its corresponding pad.
bReverse	Uses the reverse orientation of all test points.
bAxis	Checks if through holes are present. This is useful when a test point needs to be moved from top to bottom or vice versa, enabling the test point to be swapped top to bottom or vice versa.

Exceptions:

AbortException

void utestProbeMapping (boolean *bProbeMapping*)

Sets parameter to dialog Utest

Parameters:

bProbeMapping sets to map the test points to the grid of the electrical test machine.

void utestTestpoints (boolean *bTestPoints*,
int *iLoop*,
boolean *bUseMasks*,
boolean *bProbeSwaping*,
boolean *bHandlePaintedPads*,
boolean *bCircuitryCheck*,
boolean *bFilterCopperAreas*,
boolean *bViaOfSMDs*,
boolean *bDrillsWithoutPad*
)

Sets parameters to dialog Utest to section Testpoints

Parameters:

bTestPoints	if true, calculate the test points of a job and to create one or two test point layers for the job.
iLoop	sets how to test pads in a loop
bUseMasks	if true, takes all solder mask layers into account for test point calculation.
bProbeSwaping	if true, marks all test points that can be technically tested on the other side of the pcb.
bHandlePaintedPads	if thue, handle painted pads
bCircuitryCheck	if true,enables the generation of test points according to electrical test Optimization Rules.
bFilterCopperAreas	if true, reduces the number of test points generated in large coppers by removing all unnecessary test points that are satisfactorily surrounded by copper.
bViaOfSMDs	if true, generates the test points only on the via holes of SMD's, according to the attribute settings for uVia.
bDrillsWithoutPad	if true, generates test points on drill holes without pad.

void utestTestpointsBOT (boolean *bPointsBot1*,
boolean *bPointsBot2*,
boolean *bPointsBot3*,
boolean *bPointsBot4*,
boolean *bPointsBot5*,
boolean *bPointsBot6*,
boolean *bPointsBot7*
)

Sets parameters to dialog Utest to section Testpoints - Optimization Rule bottom side

Parameters:

bPointsBot1 Sets a test point on a pad that is a drilled pad and that is not connected with any track (on the top or bottom layer).

bPointsBot2 Sets a test point on a pad that is a drilled pad and that is connected with only one track on the test point side and not connected with any track on the opposite layer.

bPointsBot3 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with only one track on the opposite layer.

bPointsBot4 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to one inner layer.

bPointsBot5 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with more than one track on the opposite layer.

bPointsBot6 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to two or more inner layers.

bPointsBot7 Sets a test point on a pad that is not a drilled pad and that is connected with more than one track.

void utestTestpointsTOP (boolean *bPointsTop1,*
boolean *bPointsTop2,*
boolean *bPointsTop3,*
boolean *bPointsTop4,*
boolean *bPointsTop5,*
boolean *bPointsTop6,*
boolean *bPointsTop7*
)

Sets parameters to dialog Utest to section Testpoints - Optimization Rule top side

Parameters:

bPointsTop1 Sets a test point on a pad that is a drilled pad and that is not connected with any track (on the top or bottom layer).

bPointsTop2 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with only one track on the opposite layer.

bPointsTop3 Sets a test point on a pad that is a drilled pad and that is connected with only one track on the test point side and not connected with any track on the opposite layer.

bPointsTop4 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to one inner layer.

bPointsTop5 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side and connected with more than one track on the opposite layer.

bPointsTop6 Sets a test point on a pad that is a drilled pad and that is not connected with any track on the test point side or the opposite layer, but is connected to two or more inner layers.

bPointsTop7 Sets a test point on a pad that is not a drilled pad and that is connected with more than one track.

void validateInvalidArcs ( )

Validate Invalid Arcs

void viewAmbiguous ( )

View ambiguous contours errors

```
void viewGrid ( boolean  bVisible,
                double   ptOrigin_x,
                double   ptOrigin_y,
                double   dXStep,
                double   dYStep,
                boolean  bCross
              )
```

View Grid show or hide the grid with given parameters

Parameters:

bVisible `true` makes the grid visible; `false` hides the grid
ptOrigin_x (X coordinate) the grid origin **Point**
ptOrigin_y (Y coordinate) the grid origin **Point**
dXStep the X distance distance between grid crosses or lines
dYStep the Y distance distance between grid crosses or lines
bCross `true` show the grid crosses; `false` show the grid lines

```
void viewGrid ( boolean  bVisible,
                Point    ptOrigin,
                double   dXStep,
                double   dYStep,
                boolean  bCross
              )
```

View Grid show or hide the grid with given parameters

Parameters:

bVisible `true` makes the grid visible; `false` hides the grid
ptOrigin the grid origin **Point**
dXStep the X distance distance between grid crosses or lines
dYStep the Y distance distance between grid crosses or lines
bCross `true` show the grid crosses; `false` show the grid lines

```
void viewGrid ( boolean  bVisible,
                Point    ptOrigin,
                Point    ptStep,
                boolean  bCross
              )
```

View Grid show or hide the grid with given parameters

Parameters:

bVisible `true` makes the grid visible; `false` hides the grid
ptOrigin the grid origin **Point**
ptStep the **Point** with coordinates presenting the X and Y distance between grid crosses or lines
bCross `true` show the grid crosses; `false` show the grid lines

void viewGrid (boolean *bVisible*)

View Grid show or hide the grid

Parameters:

bVisible `true` makes the grid visible; `false` hides the grid

void viewGrid ( )

View Grid makes the grid visible

void viewGuide ( )

View guide

void viewHistory ( )

View previous display (history)

void viewInBlocks (boolean *trueFalse*)

Set/Reset Show In Blocks toggle

Parameters:

trueFalse Value of toggle

void viewMessages ( )

View messages

void viewMode (String *sMode*)

View Mode

Parameters:

sMode filled, skeleton or outline

void viewModeFilled ( )

View Mode Filled

void viewModeOutline ( )

View Mode Outline

void viewModeSkeleton ( )

View Mode Skeleton

void viewNumbers ( )

View numbers

void viewObjects (boolean *trueFalse*)

Set/Reset Show Objects toggle

Parameters:

trueFalse Value of toggle

void viewPan (double *p1_x*,
double *p1_y*,
double *p2_x*,
double *p2_y*
)

Pan main window using board coordinates

Parameters:

p1_x (X coordinate) start point of pan offset
p1_y (Y coordinate) start point of pan offset
p2_x (X coordinate) end point of pan offset
p2_y (Y coordinate) end point of pan offset

void viewPan (Point *p1*,
Point *p2*
)

Pan main window using board coordinates

Parameters:

p1 start point of pan offset
p2 end point of pan offset

void viewRefPoints (boolean *trueFalse*)

Set/Reset Show Reference Points toggle

Parameters:

trueFalse Value of toggle

void viewRepaint ( )

Repaint screen view

void viewWarning (String *message*)

View warning message

Parameters:

message Message text to be displayed

void viewZero (boolean *trueFalse*)

Set/Reset Show DPF Zero toggle

Parameters:

trueFalse Value of toggle

void viewZoom (String *sZoom*)

Zoom main window using predefined zoom levels

Parameters:

sZoom "total", "in2x", or "out2x"

void viewZoomIn ( )

Zoom main window using In 2x zoom level

void viewZoomOut ( )

Zoom main window using Out 2x zoom level

void viewZoomSelections ( )

Zoom main window on selections

void viewZoomTotal ( )

Zoom main window using total zoom level

void viewZoomWindow ( )

The command starts drag rubberband for definition of zoom window. If the window is defined the view is zoomed according to the window.

void viewZoomWindow (double *rect_xmin*,
double *rect_ymin*,
double *rect_xmax*,
double *rect_ymax*,
boolean *bScreenCenter*
)

Zoom main window using board coordinates

Parameters:

rect_xmin	(left boundary of rectangle) zoom area
rect_ymin	(bottom boundary of rectangle) zoom area
rect_xmax	(right boundary of rectangle) zoom area
rect_ymax	(top boundary of rectangle) zoom area
bScreenCenter	- true means Screen Center, false means "Center" coordinates from the Numbers Dialog

void viewZoomWindow (Rectangle *rect*,
boolean *bScreenCenter*
)

Zoom main window using board coordinates

Parameters:

rect	zoom area
bScreenCenter	- true means Screen Center, false means "Center" coordinates from the Numbers Dialog

void viewZoomWindow (double *rect_xmin*,
double *rect_ymin*,
double *rect_xmax*,
double *rect_ymax*
)

Zoom main window using board coordinates

Parameters:

rect_xmin	(left boundary of rectangle) zoom area
rect_ymin	(bottom boundary of rectangle) zoom area
rect_xmax	(right boundary of rectangle) zoom area
rect_ymax	(top boundary of rectangle) zoom area

void viewZoomWindow (Rectangle *rect*)

Zoom main window using board coordinates

Parameters:

rect zoom area

**void xmlAdd (String *sDataName,*
String *sElementName,*
String *sContent,*
ObjectList *attrArray*
)**

Adds simple element to the data section.

Parameters:

sDataName	A data section name
sElementName	Simple element name
sContent	A value (content) of the element
attrArray	Object List with pairs (attribute name, value) Eg. [{[{"name","name_1"}], [{"company","name_2"}]}]

**void xmlAdd (String *sDataName,*
String *sElementName,*
String *sContent*
)**

Adds simple element to the data section.

Parameters:

sDataName	A data section name
sElementName	Simple element name
sContent	A value (content) of the element

**void xmlAdd (String *sElementName,*
String *sContent*
)**

Adds Element to the root of the document. It is a simple element in XML document

Parameters:

sElementName	Simple element name
sContent	A value (content) of the element

**void xmlAddData (String *sParentDataName,*
String *sDataName*

)

Adds data section to the given parent data section

Parameters:

sParentDataName A parent data section name
sDataName A data section name to be added to the data section

void xmlAddData (String *sDataName*)

Adds data section to the root of the XML document

Parameters:

sDataName A data section name to be added to the XML document

void xmlCreateData (String *sDataName*)

Creates data section with a given name. Each section must be created first, otherwise it is not possible to ad any data(elements) to it

Parameters:

sDataName New data section name

void xmlDocument (String *rootName*)

Creates new XML document with given root name

Parameters:

rootName A root element (section) name

void xmlSave (String *sDestFilePath*)

Saves current XML document. It is usually the last command in a script.

Parameters:

sDestFilePath destination full file path (with file name and extension) where the current XML file will be stored.

Exceptions:

FileNotFoundException When XML file couldn't be created
IOException When I/O issue turns up during the XML file output

void YachiyoAOI_clearOutput (String *name*)

Delete generated data for the given job This simply removes the whole "name" sub-folder from the output folder Ex: YachiyoAOI_clearOutput("job1"); Ex: YachiyoAOI_clearOutput("job1");

Parameters:

name

```
void YachiyoAOI_defineArea ( int    grpIndex,
                             int    areaIndex,
                             double pos_x,
                             double pos_y
                           )
```

(Re-)define area of a group The group (given by grpIndex) must already exist. Areas must be defined in increasing order (and each defined group shall finally have at least one area). Position specifies the final location of the area, i.e. where the rectangle defined in the corresponding group is placed If the area with the given areaIndex already exists, it is repositioned accordingly Ex: YachiyoAOI_defineArea(1, 1, **Point**(24.5, 14.1)); Ex: YachiyoAOI_defineArea(1, 1, **Point**(24.5, 14.1));

Parameters:

grpIndex
areaIndex
pos_x (X coordinate)
pos_y (Y coordinate)

```
void YachiyoAOI_defineArea ( int   grpIndex,
                             int   areaIndex,
                             Point pos
                           )
```

(Re-)define area of a group The group (given by grpIndex) must already exist. Areas must be defined in increasing order (and each defined group shall finally have at least one area). Position specifies the final location of the area, i.e. where the rectangle defined in the corresponding group is placed If the area with the given areaIndex already exists, it is repositioned accordingly Ex: YachiyoAOI_defineArea(1, 1, **Point**(24.5, 14.1)); Ex: YachiyoAOI_defineArea(1, 1, **Point**(24.5, 14.1));

Parameters:

grpIndex
areaIndex
pos

```
void YachiyoAOI_defineGroup ( int    index,
                              double area_xmin,
                              double area_ymin,
                              double area_xmax,
                              double area_ymax,
                              String types
                            )
```

Start definition of a group The group will cover the provided area and allow masks of given types The types string encode both allowed types of masks (DRC, D2D, and/or CRF) and also their parameters (paraX) and optional integer value (used for passing values of additional toggles for D2D and CRF types) Group must be defined in increasing order, i.e. group 2 after group 1 etc. This call can be also used for re-definition of existing group, in that case the given group is re-set to newly provided parameters and all its masks and areas are deleted Ex: YachiyoAOI_defineGroup(1, **Rectangle**(0.0, 0.0, 115.5, 118.0), "DRC=para1,D2D=para2:1,CRF=para1:0"); Ex: YachiyoAOI_defineGroup(1, **Rectangle**(0.0, 0.0, 115.5, 118.0), "DRC=para1,D2D=para2:1,CRF=para1:0"); YachiyoAOI_defineGroup(2, **Rectangle**(0.0, 0.0, 10.3,

20.3), "D2D=para3:0,CRF=para2:1"); YachiyoAOI_defineGroup(2, **Rectangle**(0.0, 0.0, 10.3, 20.3), "D2D=para3:0,CRF=para2:1");

Parameters:

index
area_xmin (left boundary of rectangle)
area_ymin (bottom boundary of rectangle)
area_xmax (right boundary of rectangle)
area_ymax (top boundary of rectangle)
types

void YachiyoAOI_defineGroup (int *index,*
Rectangle *area,*
String *types*
)

Start definition of a group The group will cover the provided area and allow masks of given types The types string encode both allowed types of masks (DRC, D2D, and/or CRF) and also their parameters (paraX) and optional integer value (used for passing values of additional toggles for D2D and CRF types) Group must be defined in increasing order, i.e. group 2 after group 1 etc. This call can be also used for re-definition of existing group, in that case the given group is re-set to newly provided parameters and all its masks and areas are deleted Ex: YachiyoAOI_defineGroup(1, **Rectangle**(0.0, 0.0, 115.5, 118.0), "DRC=para1,D2D=para2:1,CRF=para1:0"); Ex: YachiyoAOI_defineGroup(1, **Rectangle**(0.0, 0.0, 115.5, 118.0), "DRC=para1,D2D=para2:1,CRF=para1:0"); YachiyoAOI_defineGroup(2, **Rectangle**(0.0, 0.0, 10.3, 20.3), "D2D=para3:0,CRF=para2:1"); YachiyoAOI_defineGroup(2, **Rectangle**(0.0, 0.0, 10.3, 20.3), "D2D=para3:0,CRF=para2:1");

Parameters:

index
area
types

void YachiyoAOI_defineMask (int *grpIndex,*
int *maskIndex,*
double *area_xmin,*
double *area_ymin,*
double *area_xmax,*
double *area_ymax,*
String *type*
)

(Re-)define mask of a group The group (given by grpIndex) must already exist. Masks must be defined in increasing order **Rectangle** of the mask must be inside rectange of the specified group and also its type must be allowed by the group. Mask can have multiple types, their names are then concatenated by '+' If the mask with the given maskIndex already exists, it is replaced according to the new definition Ex: YachiyoAOI_defineMask(1, 3, **Rectangle**(36.54, 56.23, 43.21, 63.93), "CRF+DRC"); Ex: YachiyoAOI_defineMask(1, 3, **Rectangle**(36.54, 56.23, 43.21, 63.93), "CRF+DRC");

Parameters:

grpIndex
maskIndex
area_xmin (left boundary of rectangle)
area_ymin (bottom boundary of rectangle)

area_xmax (right boundary of rectangle)
area_ymax (top boundary of rectangle)
type

void YachiyoAOI_defineMask (int *grpIndex,*
int *maskIndex,*
Rectangle *area,*
String *type*
)

(Re-)define mask of a group The group (given by grpIndex) must already exist. Masks must be defined in increasing order **Rectangle** of the mask must be inside rectange of the specified group and also its type must be allowed by the group. Mask can have multiple types, their names are then concatenated by '+' If the mask with the given maskIndex already exists, it is replaced according to the new definition Ex: YachiyoAOI_defineMask(1, 3, **Rectangle**(36.54, 56.23, 43.21, 63.93), "CRF+DRC"); Ex: YachiyoAOI_defineMask(1, 3, **Rectangle**(36.54, 56.23, 43.21, 63.93), "CRF+DRC");

Parameters:
grpIndex
maskIndex
area
type

void YachiyoAOI_finish ( )

Finish Yachiyo AOI, clean-up related data structures Should be called at the end of processing. Note that generated data are not removed (use **YachiyoAOI_clearOutput()** for that) Ex: **YachiyoAOI_finish()**; Ex: **YachiyoAOI_finish()**;

boolean YachiyoAOI_generateCalibration (String *name,*
double *pos_x,*
double *pos_y,*
double *startRes,*
double *endRes,*
double *step,*
String *options*
)

Generate set of images for calibration The images will be generated into the folder OUTPUTFOLDER/name, centered at given pos and with sizes given by RESOLUTION_BMP_SIZE. The first image will have resolution startRes, the next one startRes + step, and so on until the resolution endRes is met. All resolutions are in um/px and they should be multiplies of 10nm (i.e. 0.01), otherwise rounding is applied. For list of available options see **YachiyoAOI_generateOutput()** above Ex: YachiyoAOI_generateCalibration("job1-calibration1", **Point**(11.65, 12.65), 1.5, 1.8, 0.01, ""); Ex: YachiyoAOI_generateCalibration("job1-calibration1", **Point**(11.65, 12.65), 1.5, 1.8, 0.01, ""); YachiyoAOI_generateCalibration("job1-calibration2", **Point**(11.65, 12.65), 1.0, 20.0, 1.0, "MIRRORX,MIRRORY,SCALEY=0.998,REV,RLE"); YachiyoAOI_generateCalibration("job1-calibration2", **Point**(11.65, 12.65), 1.0, 20.0, 1.0, "MIRRORX,MIRRORY,SCALEY=0.998,REV,RLE");

Parameters:
name
pos_x (X coordinate)

pos_y (Y coordinate)
startRes
endRes
step
options

boolean YachiyoAOI_generateCalibration (String *name,*
Point *pos,*
double *startRes,*
double *endRes,*
double *step,*
String *options*
)

Generate set of images for calibration The images will be generated into the folder OUTPUTFOLDER/name, centered at given pos and with sizes given by RESOLUTION_BMP_SIZE. The first image will have resolution startRes, the next one startRes + step, and so on until the resolution endRes is met. All resolutions are in um/px and they should be multiplies of 10nm (i.e. 0.01), otherwise rounding is applied. For list of available options see **YachiyoAOI_generateOutput()** above Ex: YachiyoAOI_generateCalibration("job1-calibration1", **Point**(11.65, 12.65), 1.5, 1.8, 0.01, ""); Ex: YachiyoAOI_generateCalibration("job1-calibration1", **Point**(11.65, 12.65), 1.5, 1.8, 0.01, ""); YachiyoAOI_generateCalibration("job1-calibration2", **Point**(11.65, 12.65), 1.0, 20.0, 1.0, "MIRRORX,MIRRORY,SCALEY=0.998,REV,RLE"); YachiyoAOI_generateCalibration("job1-calibration2", **Point**(11.65, 12.65), 1.0, 20.0, 1.0, "MIRRORX,MIRRORY,SCALEY=0.998,REV,RLE");

Parameters:
name
pos
startRes
endRes
step
options

boolean YachiyoAOI_generateOutput (String *name,*
String *lens,*
String *fixture,*
String *options*
)

Generate complete data (set of inf files and bitmaps) for AOI as specified by Yachiyo All data (i.e. yachiyo.inf file, cadrefpointX.bmp bitmaps for reference points and group-related data (yachiyo_rip.inf and many XXXXYYYY.raw tiles in sub-folders 1, 2, etc.)) are generated into the folder OUTPUTFOLDER/name based on provided lens and fixture parameters (strings choosing appropriate LENSx and FIXTUREx parameters from settings.ini) Available options (case sensitive, multiple options can be specified in any order, but they must be separated by comma and with no spaces): REV ... generate reverse output (i.e. copper as white and background as black) MIRRORX ... mirror the layer horizontally MIRRORY ... mirror the layer vertically SCALEX=float ... scale (distort) the layer in horizontal direction by the given factor (which should be near 1.0) SCALEY=float ... same for vertical direction BMP ... generate images as uncompressed 8-bit BMPs (https: RLE ... generate images as BMPs with RLE8 compression (https: RAW ... generate images in Yachiyo specific format (currently head-less stream of data with RLE8 encoding) PBM ... generate images as "ASCII" PBMs (type P1, http: Ex: YachiyoAOI_generateOutput("job1", "x10.0", "9inch", ""); Ex: YachiyoAOI_generateOutput("job1", "x10.0", "9inch", ""); YachiyoAOI_generateOutput("job2", "x5.0", "5inch", "SCALEX=1.01,MIRRORY,REV,RLE"); YachiyoAOI_generateOutput("job2", "x5.0", "5inch", "SCALEX=1.01,MIRRORY,REV,RLE");

Parameters:

name
lens
fixture
options

String YachiyoAOI_getStrings (String *kind*)

Return all possible values for the given GUI component (typically labels for items of comboboxes) The values are returned in one string, separated by "|" Kind must be one of the following strings: DRC, D2D, CRF, LENS, RESOLUTION, FIXTURE Ex: YachiyoAOI_getStrings("FIXTURE"); -> "5inch|9inch|13inch|125mm" Ex: YachiyoAOI_getStrings("FIXTURE"); -> "5inch|9inch|13inch|125mm"

Parameters:

kind

boolean YachiyoAOI_init (String *iniFile*)

Initialize Yachiyo AOI, parse provided settings.ini Must be called first before any other YachiyoAOI command is used Ex: YachiyoAOI_init("H:/UcamBugs/Yachiyo/WORK/YAOI_params/settings.ini"); Ex: YachiyoAOI_init("H:/UcamBugs/Yachiyo/WORK/YAOI_params/settings.ini");

Parameters:

iniFile

void YachiyoAOI_reset ( )

Delete generated data for the given job This simply removes the whole "name" sub-folder from the output folder Ex: YachiyoAOI_clearOutput("job1");

void YachiyoAOI_setRefPoint (int *index,*
double *pos_x,*
double *pos_y*
)

Set reference point to given pos Points can be set in random order This call can be also used for re-position of any already defined point Ex: YachiyoAOI_setRefPoint(1, **Point**(563.35, 12.65)); Ex: YachiyoAOI_setRefPoint(1, **Point**(563.35, 12.65));

Parameters:

index
pos_x (X coordinate)
pos_y (Y coordinate)

void YachiyoAOI_setRefPoint (int *index,*
Point *pos*
)

Set reference point to given pos Points can be set in random order This call can be also used for re-position of any already defined point Ex: YachiyoAOI_setRefPoint(1, **Point**(563.35, 12.65)); Ex: YachiyoAOI_setRefPoint(1, **Point**(563.35, 12.65));

Parameters:

index

pos

Variable Documentation

int FILE_ATTRIBUTES = 2

File info field containing file/dir attributes

int FILE_MODIFICATION_DATE = 4

File info field containing last modification time

int FILE_NAME = 5

File info field containing file/dir name

int FILE_PARENT = 1

File info field containing file/dir parent directory

int FILE_SIZE = 3

File info field containing file/dir size

int FILE_TYPE = 0

File info field containing type information

int LAYER_ACTIVITY = 5

Layer info field containing layer's activity

int LAYER_APERTURES = 6

Layer info field containing layer's activity

int LAYER_ATTACH = 3

Layer info field containing layer's attachment

int LAYER_CLASS = 1

Layer info field containing layer's class

int LAYER_INDEX = 4

Layer info field containing layer's index

int LAYER_NAME = 0

Layer info field containing layer's name

int LAYER_SUBCLASS = 2

Layer info field containing layer's subclass

Generated on Wed Mar 21 04:37:34 2018 for Visual HyperScript API Specification by  *1.5.4*

Arc Class Reference

List of all members.

Public Attributes

- **Point cp**

arc center point

- **Point fp**

arc start point

- String **sense**

arc sense

- **Point tp**

arc end point

Detailed Description

arc representation

Member Data Documentation

Point Arc::cp

arc center point

Point Arc::fp

arc start point

String Arc::sense

arc sense

Point Arc::tp

arc end point

Line Class Reference

List of all members.

Public Attributes

- **Point fp**

line start point

- **Point tp**

line end point

Detailed Description

line representation

Member Data Documentation

Point Line::fp

line start point

Point Line::tp

line end point

Generated on Wed Mar 21 04:37:34 2018 for Visual HyperScript API Specification by doxygen *1.5.4*

Point Class Reference

List of all members.

Public Attributes

- double **x**

x coordinate of the point

- double **y**

y coordinate of the point

Detailed Description

point representation

Member Data Documentation

double Point::x

x coordinate of the point

double Point::y

y coordinate of the point

Generated on Wed Mar 21 04:37:34 2018 for Visual HyperScript API Specification by doxygen *1.5.4*

Rectangle Class Reference

List of all members.

Public Attributes

- double **xmax**

right rectangle boundary

- double **xmin**

left rectangle boundary

- double **ymax**

top rectangle boundary

- double **ymin**

bottom rectangle boundary

Detailed Description

rectangle representation

Member Data Documentation

double Rectangle::xmax

right rectangle boundary

double Rectangle::xmin

left rectangle boundary

double Rectangle::ymax

top rectangle boundary

double Rectangle::ymin

bottom rectangle boundary

Generated on Wed Mar 21 04:37:34 2018 for Visual HyperScript API Specification by doxygen *1.5.4*

Deprecated List

Member **compareNet**

use **compareNet(boolean, boolean, boolean, boolean, boolean, boolean, boolean, double, boolean, double, String, boolean, boolean)**

use **compareNet(boolean, boolean, boolean, boolean, boolean, boolean, boolean, double, boolean, double, String, boolean, boolean)**

Member **copperCount**

Copper count without mask layer usage

Parameters:

sOpt Set to "job", "layer", or "inner"

Member **deselectObjectAttribute**

deselectObjectAttribute deselect objects with attribute with the given name and value from current job.

Parameters:

sAttrName The object attribute name

sAttrValue The object attribute value

Member **deselectObjectAttribute**

deselectObjectAttribute deselect objects with attribute with the given name from current job.

Parameters:

sAttrName The object attribute name

Member **insertPolydrawRect**

Insert polydraw rectangle using current aperture

Parameters:

p1 bottom left point of the rectangle

p2 top right point of the rectangle

bRectCW set true if the rectangle should be CW

bSel set true if the rectagle should be selected

Member **selectByApertureShape**

Select or deselect all objects of the specified object type.

Parameters:

selectMode Either + (select) or - (deselect)

apertureShapes Comma separated list of cir, don, rec, squ, box, oct, com, the, con, tex, blo

Member **selectByAttributeName**

Select or deselect all objects of the specified attribute name.

Parameters:

selectMode Either + (select) or - (deselect)

sName - attribute name

Member **selectByAttributeValue**

Select or deselect all objects of the specified attribute value.

Parameters:

selectMode Either + (select) or - (deselect)
sName - attribute name
sValue - attribute value

Member **selectByObjectType**

Select or deselect all objects of the specified object type.

Parameters:

selectMode Either + (select) or - (deselect)
objectTypes Comma separated list of f (flash), d (draw), a (arc) or v (vector text)

Member **selectObjectAttribute**

selectObjectAttribute Select objects with set attribute with the given name and value from current job.

Parameters:

sAttrName The object attribute name
sAttrValue The object attribute value

Member **selectObjectAttribute**

selectObjectAttribute Select objects with set attribute with the given name from current job.

Parameters:

sAttrName The object attribute name

Member **setAttributeOnObject**

This function is GUI ONLY, replaced by addObjectAttribute(sAttrName, sAttrValue)

See also:

addObjectAttribute(String sAttrName, String sAttrValue) Insert attribute on objects

Parameters:

attrName Name of attribute
attrValue Value of attribute

Arc Member List

This is the complete list of members for **Arc**, including all inherited members.

cp	**Arc**
fp	**Arc**
sense	**Arc**
tp	**Arc**

Generated on Wed Mar 21 04:37:34 2018 for Visual HyperScript API Specification by  *1.5.4*