OrCAD X Capture TCL Solutions

Product Version 23.1
September 2023

Contents

1

Automating Common OrCAD Tasks using TCL Scripts

As the design complexity increases, many of the tasks done when creating a stable and firm design get repetitive. OrCAD® Capture provides TCL language support to automate the repetitive tasks.

This document covers some of the common scenarios that can benefit from using TCL scripts. You can use these scripts directly, or with some changes related to system or design to complete the task.

This section has scripts for completing the following tasks:

- Adding a New Property to a Selected Part
- Modifying Pin Name or Number in a library
- Verifying the Selected Object as a Library or Not
- Creating Capture CIS BOM from the Windows Command Prompt
- Defining a property for selected occurrence parts
- Filtering Net Alias on an Active Schematic Page
- Getting Library Part User properties
- Getting Pin Names from a Library Part
- Getting the Occurrence Object of a Selected Part
- Listing Properties of the Selected Object in a Schematic Page
- Selecting a Net in a Schematic Page using Net Name
- Setting the Implementation Type on a Part to a Specfic Value
- Displaying Selected Objects Name and Location
- Modifying Selected Wire Color and Width
- Adding Property to a Selected Wire

Adding a New Property to a Selected Part

This script adds a new property to a selected part placed on a schematic page. Using the script you can

- Add a new property and set it's value for a selected part,
- Display the name and value of the property. If the property already exists on the selected part, only property's name is displayed.

To add a new property, perform the following steps in the Capture command window:

1. Select a part to add a new property
2. Source the TCL file, for example `source {D:/AddDispProp.tcl}`
3. Run `AddDisplayProperty "<New Property Name>" "<New Property Value>"`
   For example:
   `AddDisplayProperty "P1" "1"` where `"P1"` is property name and `"1"` is property value.

AddDispProp.tcl

```
proc ConvertUserToDoc { pPage pUser } {
 set lDocDouble [expr "[$pPage GetPhysicalGranularity] * $pUser + 0.5"]
 set lDoc [expr "round($lDocDouble)"]
 return $lDoc
}
proc AddDisplayProperty {pProp pVal} {
    # Get the selected objects
    set lSelObjs1 [GetSelectedObjects]
    set lObj1 [lindex $lSelObjs1 0]
    set lPropNameCStr [DboTclHelper_sMakeCString $pProp]
    set lPropValueCStr [DboTclHelper_sMakeCString $pVal]
    set lStatus [$lObj1 SetEffectivePropStringValue $lPropNameCStr $lPropValueCStr]
    set varNullObj NULL
    set pDispProp [$lObj1 GetDisplayProp $lPropNameCStr $lStatus]
    set lStatus [DboState]
    if { $pDispProp == $varNullObj } {
        set rotation 0
        set logfont [DboTclHelper_sMakeLOGFONT]
        set color $::DboValue_DEFAULT_OBJECT_COLOR
        #set displocation [DboTclHelper_sMakeCPoint [expr $xlocation] [expr $ylocation]]
        if {[catch {set lPickPosition [GetLastMouseClickPointOnPage]} lResult] } {
            set lX 0
            set lY 0
            set displocation [DboTclHelper_sMakeCPoint $intX $intY]
        } else {
            set page [$lObj1 GetOwner]
            set lX [ConvertUserToDoc $page [lindex $lPickPosition 0]]
            set lY [ConvertUserToDoc $page [lindex $lPickPosition 1]]
            set displocation [DboTclHelper_sMakeCPoint $lX $lY]
        }
        set pNewDispProp [$lObj1 NewDisplayProp $lStatus $lPropNameCStr $displocation $rotation
$logfont $color]
        #DO_NOT_DISPLAY            = 0,
        #VALUE_ONLY                = 1,
        #NAME_AND_VALUE            = 2,
        #NAME_ONLY                = 3,
        #BOTH_IF_VALUED            = 4,
        $pNewDispProp SetDisplayType $::DboValue_NAME_AND_VALUE
    } else {
        # 0 = DoNotDisplay, 1 = Value, 2 = Name and Value, 3 = Name, 4 = If Value Exists
        $pDispProp SetDisplayType $::DboValue_NAME_ONLY
    }
}
```

Modifying Pin Name or Number in a library

The following script changes the pin name and pin number of `GND` pin to new name. To run the script, do the following:

1. Open a design in Capture.
2. Open any schematic page in the design
3. Select a part that has GND pin
4. Open Capture command window, if not opened already.
5. Source the following TCL script in the Capture command window, for example `source {D:\setNewPinName.tcl}`
6. Check the properties of the GND pin . The pin name changes to `GND_NEW` and pin number changes to `100`.
   If you want to set different values for the pin name and pin name, modify the script before sourcing it in the command window.

setNewPinName.tcl

```
#This TCL file updates the pin name and pin number
#of a GND pin of single selected object to GND_NEW and 100 respectively.
set lStatus [DboState]
set lNullObject NULL
set lObject [GetSelectedObjects]
set lPackage [$lObject GetPackage $lStatus]

# Get the first device in the package
# if you need to change for specific device
# you can get the device using device designator.
set lDevice [$lPackage GetDevice 0 $lStatus]
if { $lDevice != $lNullObject } {

    set lCell [$lDevice GetCell $lStatus]

    #Gets the normal part from the device
    set lNormalPart [$lCell FindPart 0 $lStatus]


    # Since pin location is stored in symbol pin
    # we need to find the symbol pin object for the pin
    # whose pin number we want to change.

    set lLibPartPinIter [$lNormalPart NewLPinsIter $lStatus]
    set lSymbolPin [$lLibPartPinIter NextPin $lStatus]
```

```
     set lSymbolPin [$lLibPartPinIter NextPin $lStatus]
     set lPinName [DboTclHelper_sMakeCString]
     while {$lSymbolPin != $lNullObject} {
         $lSymbolPin GetPinName $lPinName
         set lPinNameTcl [DboTclHelper_sGetConstCharPtr $lPinName]
         if {$lPinNameTcl=="GND"} {

    #Sets the GND pin to GND_NEW pin
             set lNewName [DboTclHelper_sMakeCString "GND_NEW"]
             $lSymbolPin SetPinName $lNewName


             # If symbol pin found
             # Get the pin position
             set lPinPosition [ $lSymbolPin GetPinPosition $lStatus]

             # Sets new pin number 100
             set lNewPinNumber [DboTclHelper_sMakeCString "100"]
             set lPinPosition [DboTclHelper_sMakeInt $lPinPosition]

             # Changes the pin number
             $lDevice NewPinNumber $lNewPinNumber $lPinPosition 1 -1

             puts "Pin name and Pin number changed successfully"

             break
         }
         set lSymbolPin [$lLibPartPinIter NextPin $lStatus]
     }
 }
```

Verifying the Selected Object as a Library or Not

Use the following script to check if the object currently selected in the Project Manager window is a library or not. Source the script in Capture using the following template in the Capture command window:

```
source {<scriptname.tcl>}
```

libOrNot.tcl

```
#Code to check if selected object in Project Manager is library or not
set lLibName [GetSelectedPMItems]
set lLibName [ string map { "{" "" "}" ""} $lLibName ]
set lOlbIndex [string first ".olb" $lLibName]
if { $lOlbIndex != -1 } {
puts "It is a Library"

} else {
puts "It is not Library"

}
```

Creating Capture CIS BOM from the Windows Command Prompt

This script does the following:

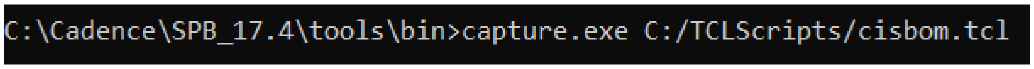
1. Opens and loads the design file (`bench_allegro.dsn`), which can be found at `tools/capture/samples/Schematic/Bench`.
   The design file used as a sample for this script.
2. Creates Capture CIS Bill of Materials (BOM) using the XML dialog settings for the Standard Bill of Materials dialog box.
   You need to have XML dialog settings file at the specified path to run the script. The XML dialog settings template is mentioned in <section>
3. Saves the generated BOM as `TEST.BOM` at the specified path.
4. Exits the design and Capture.

To run the following script at the Windows command prompt, do the following:

1. Save the TCL script, named `cisbom.tcl` on your system.
2. Open the Windows command prompt.
3. Now run the command:*`<path to the Capture.exe file> <path to the TCL script saved`*

in step 1>

For example:



```
C:\Cadence\SPB_17.4\tools\bin>capture.exe C:/TCLScripts/cisbom.tcl
```

cisbom.tcl

```
#open and load the Capture CIS design
Menu "File::Open::Design" | FileDialog "OK"
"D:/Cadence/tools/capture/samples/Schematic/Bench/bench_allegro.DSN" 1
capDesignOpenMessage
capEventNotify 4004 [capGetPMId]

#create CIS BOM using xml dialog settings
SelectPMItem "./bench_allegro.DSN "
Menu "Reports::CIS Bill of Materials::Standard" | DialogBox "OK"
"D:/Standard_Bill_of_Materials.xml"

#save the generated bom as TEST.BOM
MenuCommand 57927 | MessageBox "YES" "Save changes to D:/Cadence/spb" | FileDialog "OK"
"D:/TEST.BOM" 1

#exit capture
SelectPMItem "./bench_allegro.DSN "
Menu "File::close"
Menu "File::Exit"
```

The following section describes the XML dialog settings template that is used the `cisbom.tcl` script.

Standard Bill of Materials dialog box XML dialog settings

Following is the XML dialog settings template for Standard Bill of Materials dialog box:

Standard_Bill_of_Materials.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<DialogControls>
    <Control Type="PUSH_BUTTON" Enable="TRUE" Visible="TRUE" Id="1031">
        <Value><![CDATA[&OK]]></Value>
    </Control>
    <Control Type="PUSH_BUTTON" Enable="TRUE" Visible="TRUE" Id="2">
        <Value><![CDATA[&Cancel]]></Value>
    </Control>
    <Control Type="PUSH_BUTTON" Enable="TRUE" Visible="TRUE" Id="57670">
        <Value><![CDATA[&Help]]></Value>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="-1">
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="1">
        <Value><![CDATA[&Template Name]]></Value>
    </Control>
    <Control Type="COMBO_BOX" Enable="TRUE" Visible="TRUE" Id="1017">
        <Value><![CDATA[Eng Bill Of Materials]]></Value>
        <Value><![CDATA[Eng Bill Of Materials]]></Value>
    </Control>
    <Control Type="PUSH_BUTTON" Enable="TRUE" Visible="TRUE" Id="1114">
        <Value><![CDATA[De&lete]]></Value>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="1200">
        <Value><![CDATA[Report Properties]]></Value>
    </Control>
    <Control Type="COMBO_BOX" Enable="TRUE" Visible="TRUE" Id="1030">
        <Value><![CDATA[]]></Value>
        <Value><![CDATA[ActivepartsID]]></Value>
        <Value><![CDATA[Availability]]></Value>
        <Value><![CDATA[Datasheet]]></Value>
        <Value><![CDATA[Distributor]]></Value>
        <Value><![CDATA[Distributor Part Number]]></Value>
        <Value><![CDATA[Implementation]]></Value>
        <Value><![CDATA[Layout PCB Footprint]]></Value>
        <Value><![CDATA[Manufacturer]]></Value>
        <Value><![CDATA[Manufacturer Part Number]]></Value>
        <Value><![CDATA[OrgAddr1]]></Value>
        <Value><![CDATA[OrgAddr2]]></Value>
        <Value><![CDATA[OrgAddr3]]></Value>
        <Value><![CDATA[OrgName]]></Value>
        <Value><![CDATA[Page Count]]></Value>
        <Value><![CDATA[Page Modify Date]]></Value>
        <Value><![CDATA[Page Size]]></Value>
        <Value><![CDATA[Part Type]]></Value>
        <Value><![CDATA[PCB Footprint]]></Value>
        <Value><![CDATA[Power]]></Value>
        <Value><![CDATA[Price]]></Value>
        <Value><![CDATA[Rating]]></Value>
        <Value><![CDATA[RevCode]]></Value>
        <Value><![CDATA[Schematic Part Path]]></Value>
        <Value><![CDATA[Source Library]]></Value>
        <Value><![CDATA[Source Package]]></Value>
        <Value><![CDATA[Title]]></Value>
        <Value><![CDATA[Tolerance]]></Value>
        <Value><![CDATA[Voltage]]></Value>
    </Control>
    <Control Type="LIST_BOX" Enable="TRUE" Visible="TRUE" Id="189">
        <Value><![CDATA[Item Number]]></Value>
        <Value><![CDATA[Item Number]]></Value>
        <Value><![CDATA[Quantity]]></Value>
        <Value><![CDATA[Value]]></Value>
        <Value><![CDATA[Description]]></Value>
        <Value><![CDATA[Part Number]]></Value>
        <Value><![CDATA[Part Reference]]></Value>
    </Control>
    <Control Type="PUSH_BUTTON" Enable="FALSE" Visible="TRUE" Id="1012">
        <Value><![CDATA[&Add ->]]></Value>
    </Control>
    <Control Type="PUSH_BUTTON" Enable="FALSE" Visible="TRUE" Id="1108">
        <Value><![CDATA[<- Remo&ve]]></Value>
    </Control>
    <Control Type="OWNER_DRAWN" Enable="FALSE" Visible="TRUE" Id="1036">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[IDB_UPARROW]]></Text>
    </Control>
    <Control Type="OWNER_DRAWN" Enable="FALSE" Visible="TRUE" Id="1039">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[IDB_DNARROW]]></Text>
    </Control>
    <Control Type="PUSH_BUTTON" Enable="TRUE" Visible="TRUE" Id="1266">
        <Value><![CDATA[Delete &User Property]]></Value>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="FALSE" Visible="TRUE" Id="1020">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[&Keyed]]></Text>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="TRUE" Visible="TRUE" Id="1284">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Allo&w Saving Title Block Properties]]></Text>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="FALSE" Visible="TRUE" Id="1311">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[List Relational &Data Fields]]></Text>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="1034">
        <Value><![CDATA[Part Reference Options]]></Value>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="TRUE" Visible="TRUE" Id="1182">
        <Value><![CDATA[1]]></Value>
        <Text><![CDATA[Sta&ndard]]></Text>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="TRUE" Visible="TRUE" Id="1184">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[&Standard- separate line per part]]></Text>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="TRUE" Visible="TRUE" Id="1186">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Comp&ressed]]></Text>
    </Control>
    <Control Type="COMBO_BOX" Enable="TRUE" Visible="TRUE" Id="1197">
        <Value><![CDATA[Space(' ')]]></Value>
        <Value><![CDATA[Space(' ')]]></Value>
        <Value><![CDATA[Comma(',')]]></Value>
    </Control>
    <Control Type="EDIT" Enable="TRUE" Visible="TRUE" Id="1069">
        <Value><![CDATA[]]></Value>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="-1">
        <Value><![CDATA[]]></Value>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="TRUE" Visible="TRUE" Id="1079">
        <Value><![CDATA[0]]></Value>
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Output Mechanical Part Data]]></Text>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="FALSE" Visible="TRUE" Id="1296">
        <Value><![CDATA[1]]></Value>
        <Text><![CDATA[Mechanical parts only]]></Text>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="FALSE" Visible="TRUE" Id="1185">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[&Both mechanical parts and assemblies]]></Text>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="-1">
        <Value><![CDATA[]]></Value>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="FALSE" Visible="TRUE" Id="1310">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Relational Data Displa&yed]]></Text>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="FALSE" Visible="TRUE" Id="1309">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Hori&zontal Output]]></Text>
    </Control>
    <Control Type="EDIT" Enable="FALSE" Visible="TRUE" Id="1308">
        <Value><![CDATA[1]]></Value>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="TRUE" Visible="TRUE" Id="1248">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[E&xport BOM report to Excel]]></Text>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="FALSE" Visible="TRUE" Id="1142">
        <Value><![CDATA[1]]></Value>
        <Text><![CDATA[Mer&ge BOM Reports]]></Text>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="1046">
        <Value><![CDATA[Variants]]></Value>
    </Control>
    <Control Type="AUTOCHECK_BOX" Enable="TRUE" Visible="TRUE" Id="1306">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Variant "Not Stuffed" &Qty 0 Displayed]]></Text>
    </Control>
    <Control Type="LIST_BOX" Enable="TRUE" Visible="TRUE" Id="1272">
        <Value><![CDATA[Variation #1]]></Value>
        <Value><![CDATA[<Core Design>]]></Value>
        <Value><![CDATA[Variation #1]]></Value>
        <Value><![CDATA[Variation #2]]></Value>
    </Control>
    <Control Type="GROUPBOX" Enable="TRUE" Visible="TRUE" Id="-1">
        <Value><![CDATA[Scope]]></Value>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="TRUE" Visible="TRUE" Id="1315">
        <Value><![CDATA[1]]></Value>
        <Text><![CDATA[Process Entire &Design]]></Text>
    </Control>
    <Control Type="AUTORADIO_BUTTON" Enable="TRUE" Visible="TRUE" Id="1316">
        <Value><![CDATA[0]]></Value>
        <Text><![CDATA[Process &Selection]]></Text>
    </Control>
</DialogControls>
```

Defining a property for selected occurrence parts

The script does the following:

- Get the total number of selected objects.
- Find the parent schematic's occurrence for all the objects.
- If the parent schematic occurrence is found in the list of objects, add property to the found occurrence.

setOcc

```
proc SetOcc {} {
#### Get the selected objects
set lSelObjs1 [GetSelectedObjects]
set lObj1 [lindex $lSelObjs1 0]
set objlength [llength $lSelObjs1]
set lNullObj NULL
#####For loop to search all the objects and occurrences
#####to get the selected occurrence
for {set i 0} {$i < $objlength} {incr i} {
    set obj [lindex $lSelObjs1 $i]
    set lParentSchematicOcc [GetInstanceOccurrence]
    if {$lParentSchematicOcc != $lNullObj} {
        set objOcc [$obj GetObjectOccurrence $lParentSchematicOcc]
        if { $objOcc != $lNullObj } {
   ####Adding the properties to the selected occurence
            set lPropNameCStr [DboTclHelper_sMakeCString "Color"]
            set lPropValueCStr [DboTclHelper_sMakeCString "RGB(192,192,192)"]
            set lStatus [$objOcc SetEffectivePropStringValue $lPropNameCStr $lPropValueCStr]
            set lPropNameCStr [DboTclHelper_sMakeCString "BOM_IGNORE"]
            set lPropValueCStr [DboTclHelper_sMakeCString "TRUE"]
            set lStatus [$objOcc SetEffectivePropStringValue $lPropNameCStr $lPropValueCStr]
            }
        }
    }
}
```

Source the script in Capture command window, such as `source {D:\setOcc.tcl}`.

Filtering Net Alias on an Active Schematic Page

This script finds all the net aliases on an active schematic page. To run the script in the command window:

1. Source the TCL script, such as `source {D:/filterNetAlias.tcl}`
2. Run `capGetAllAliasesOnPage [GetActivePage]`

filterNetAlias.tcl

```
proc capGetAllAliasesOnPage { pPage } {
    set lNullObj NULL
    set lStatus [DboState]
    set pWiresIter [$pPage NewWiresIter $lStatus]
    set pWire [$pWiresIter NextWire $lStatus]
    while {$pWire !=$lNullObj} {
        #WireAliases
        set pAliasIter [$pWire NewAliasesIter $lStatus]
        set pAlias [$pAliasIter NextAlias $lStatus]
        while { $pAlias!=$lNullObj} {
            puts $pAlias
            set pAlias [$pAliasIter NextAlias $lStatus]
        }
    delete_DboWireAliasesIter $pAliasIter
    $pAliasIter -delete
    unset pWire
    set pWire [$pWiresIter NextWire $lStatus]
    }
    delete_DboPageWiresIter $pWiresIter
    $pWiresIter -delete
    unset pWire
}
```

Getting Library Part User properties

This script does the following:

- Iterates through the specified library
- Finds the specified library part in the specified library
- Displays user properties on the specified library part

To use the script in Capture, do the following:

1. Source the script in Capture using command window, such as `source {D:\displyuserprop.tcl}`.
2. Use the `DisplayUserProp` command to get the user properties of a specified library part, such as `DisplayUserProp <LibName> <PartName>`.

displyuserprop.tcl

```
proc PropertyDump { pDataBaseObj } {
    set NullObject NULL
    set stateObj [DboState]
    set userPropValue [DboTclHelper_sMakeCString]
    set userPropName [DboTclHelper_sMakeCString]
    set userPropIter [$pDataBaseObj NewUserPropsIter $stateObj]
    set userProp [$userPropIter NextUserProp $stateObj]
    while {$userProp != $NullObject} {
        $userProp GetName $userPropName
        $userProp GetStringValue $userPropValue
        set propMessage "\t\tUser Property: '[DboTclHelper_sGetConstCharPtr $userPropName]'
Value: '[DboTclHelper_sGetConstCharPtr $userPropValue]"
        puts $propMessage
        set userProp [$userPropIter NextUserProp $stateObj]
    }
}
proc MakeLibPartDisplayProp {aCell}  {

    set lStatus [DboState]
    set lNullObject NULL

    set lPartsIter [$aCell NewPartsIter $lStatus]
    set lPart [$lPartsIter NextPart $lStatus]

     set lCStrPartName [DboTclHelper_sMakeCString ]
     set lCStrPropValue [DboTclHelper_sMakeCString ]
     set lPropFound 0
     while {$lPart != $lNullObject} {
        $lPart GetName $lCStrPartName
        set lPartName [DboTclHelper_sGetConstCharPtr $lCStrPartName]
        puts "Part Name:$lPartName"
        PropertyDump $lPart

        set lPart [$lPartsIter NextPart $lStatus]
    }
    delete_DboCellPartsIter $lPartsIter
    return $lPropFound
}
proc MakeLibPackageDisplayProp {aLibName aPackageName}  {
    set lSession $::DboSession_s_pDboSession
    DboSession -this $lSession
    set lStatus [DboState]
    set lNullObject NULL
    set lIsLibLoaded 1
    set lLibName [DboTclHelper_sMakeCString $aLibName]

    set lLib [$lSession GetOpenLib $lLibName $lStatus]
    if { $lLib == $lNullObject} {

        set lIsLibLoaded 0
        set lLib [$lSession GetLib $lLibName $lStatus]
    }


    if { $lLib == $lNullObject} {
        puts "Can't open library"
        return
    }

    set pkgIter [$lLib NewPackagesIter $lStatus]
    set pPkg [$pkgIter NextPackage $lStatus]
    set lNullObj NULL
    set pActualPkgName [DboTclHelper_sMakeCString]
    set pPartName [DboTclHelper_sMakeCString]
    set lFound 0
    while {$pPkg!=$lNullObj} {
        # Get Package Name
        $pPkg GetName $pActualPkgName
        set lTclPackageName [DboTclHelper_sGetConstCharPtr $pActualPkgName]
        if {$lTclPackageName==$aPackageName} {
            puts "Found Package"
            set pDeviceIter [$pPkg NewDevicesIter $lStatus]
            set DeviceNullObj NULL
            set pDevice [$pDeviceIter NextDevice $lStatus]
            while {$pDevice!=$DeviceNullObj}  {
                set cellObj [$pDevice GetCell $lStatus]
                set lFound [MakeLibPartDisplayProp $cellObj]
                set pDevice [$pDeviceIter NextDevice $lStatus]
            }
            delete_DboPackageDevicesIter $pDeviceIter
            break
        }
        set pPkg [$pkgIter NextPackage $lStatus]
    }
    delete_DboLibPackagesIter $pkgIter
    if { $lFound==1 } {
        puts "Saving the Library"
        $lSession MarkAllLibForSave $lLib
        $lSession SaveLib $lLib
    }
    if { $lIsLibLoaded == 0 } {
        $lSession RemoveLib $lLib
    }
    DboTclHelper_sReleaseAllCreatedPtrs
}
proc DisplayUserProp {aLibName aPartName}  {
```

```
proc DisplayUserProp {aLibName aPartName}  {
    MakeLibPackageDisplayProp $aLibName $aPartName
}
```

Getting Pin Names from a Library Part

You can use this script to get the following details:

- Get the pin names of a selected library added in the Project Manager's Library folder.
- Get the pin names of a selected object in a schematic page

Do the following to get pin names of a selected library part added in the Project Manager's library folder:

1. Source the TCL script file in the Capture command window, such as `source {d:/GetPinName.tcl}`
2. Run `set lObject [GetSelectedPMItems]`
3. Run `GetPinName $lObject`

Do the following to get pin names of a selected object in a schematic page:

1. Source the TCL script file in the Capture command window, such as `source {d:/GetPinName.tcl}`
2. Select an object in a schematic page
3. Run `set lObject [GetSelectedObjects]`
4. Run `set lPackage [$lObject GetPackage $lStatus]`
5. Run `GetPinName $lPackage`

GetPinName.tcl

```
proc GetPinName { pPackage } {

    set lNullObj NULL

    set lType [$pPackage GetObjectType]
    if { $lType != $::DboBaseObject_PACKAGE } {
        return
    }

    set lStatus [DboState]
    #Iterating Device
    set lDeviceIter [$pPackage NewDevicesIter $lStatus]
    set lDevice [$lDeviceIter NextDevice $lStatus]
    while {$lDevice!=$lNullObj} {

        #Getting cell from device
        set lCell [$lDevice GetCell $lStatus]

        #LibPartIter
        set lLipPartIter [$lCell NewPartsIter $lStatus]
        set lPart [$lLipPartIter NextPart $lStatus]
        while {$lPart!=$lNullObj} {

            #PinIter
            set lPinIter [$lPart NewLPinsIter $lStatus]
            set lPin [$lPinIter NextPin $lStatus]
            while {$lPin!=$lNullObj} {

                #Getting Pin Name [GetPinName :- Member Function]
                set lPinName [DboTclHelper_sMakeCString]
                $lPin GetPinName $lPinName
                set lPinNameString [DboTclHelper_sGetConstCharPtr $lPinName]
                puts "PinName :- $lPinNameString"

                #Getting Pin Name [sGetPinName :- Static Function]
                #set lPinName2 [DboSymbolPin_sGetPinName $lPin $lStatus]
                #set lPinNameString2 [DboTclHelper_sGetConstCharPtr $lPinName2]
                #puts "PinName :- $lPinNameString2"

                set lPin [$lPinIter NextPin $lStatus]

            }

            set lPart [$lLipPartIter NextPart $lStatus]

        }
        set lDevice [$lDeviceIter NextDevice $lStatus]

    }

}
```

Getting the Occurrence Object of a Selected Part

The following script does the following:

- Searches for all the objects in the in design
- Finds the occurrence object

getOcc.tcl

```
set lNullObject NULL
foreach lObject $lObjects {
 set lType [$lObject GetObjectType]
  if { $lType == $::DboBaseObject_PLACED_INSTANCE} {
   set lOcc [[GetSelectedObjects] GetObjectOccurrence [GetInstanceOccurrence]]
   set lInstOcc [DboOccurrenceToDboInstOccurrence $lOcc]
    if { $lInstOcc != $lNullObject} {
    set lPartReference [DboTclHelper_sMakeCString]
    $lInstOcc GetReferenceDesignator $lPartReference
    set lPartRefStr [DboTclHelper_sGetConstCharPtr $lPartReference]
    puts "Occ Found - $lPartRefStr"
    }
  }
}
```

⚠ You need to specify `$lObjects` parameter before you use it in the for loop.

Listing Properties of the Selected Object in a Schematic Page

This script lists all properties, including user-defined and database-based, of a object currently selected in a schematic page. To run this script, select an object present in the schematic page and source the script in the command window.

listAllProp.tcl

```
#####List All properties of an Object, including user and database
set lObject [GetSelectedObjects]
set lStatus [DboState]
set lPropsIter [$lObject NewEffectivePropsIter $lStatus]
set lNullObj NULL
set lPrpName [DboTclHelper_sMakeCString]
set lPrpValue [DboTclHelper_sMakeCString]
set lPrpType [DboTclHelper_sMakeDboValueType]
set lEditable [DboTclHelper_sMakeInt]
#get the first effective property
set lStatus [$lPropsIter NextEffectiveProp $lPrpName $lPrpValue $lPrpType $lEditable]
while {[$lStatus OK] == 1} {
    puts "Name = [DboTclHelper_sGetConstCharPtr $lPrpName]                 Value = 
[DboTclHelper_sGetConstCharPtr $lPrpValue]"
    #get the next effective property
    set lStatus [$lPropsIter NextEffectiveProp $lPrpName $lPrpValue $lPrpType $lEditable]
    }
delete_DboEffectivePropsIter $lPropsIter
```

Selecting a Net in a Schematic Page using Net Name

You can select a net in an active schematic page using a net name. To run the script, do the following in the Capture command window:

1. Source the TCL script, such as `source {D:/dispNetByName.tcl}`
2. Run `capSelectNetByName <Net Alias Name>`, such as `capSelectNetByName alias1`.

dispNetByBame.tcl

```
proc capSelectNetByName { pNetName } {
    # requires schematic view to be active
    if { [IsSchematicViewActive] != 1 } {
        set lReturnValue [list "No schematic view active"]
        return $lReturnValue
    }

    set NetName $pNetName
    set lStatus [DboState]
    set lNullObj NULL
    set lNetName [DboTclHelper_sMakeCString $NetName]
    set lPage [GetActivePage]
    set pNet [$lPage GetNet $lNetName $lStatus]
    if { $pNet!=$lNullObj } {
        DboNetWiresIter lWiresIter $pNet $::IterDefs_ALL
        set lWire [lWiresIter NextWire $lStatus]
        while {$lWire!=$lNullObj} {
        set pId [$lWire GetId $lStatus]
        SelectObjectById $pId
        set lWire [lWiresIter NextWire $lStatus]
        }
    }
}
```

Setting the Implementation Type on a Part to a Specfic Value

This script does the following things:

- Creates a new Capture Session
- Gets the Capture library
- Finds the required package in the library
  If the part is found in the package, it sets the `SetContentsViewType` property to the required implementation type.
  In this case, we have set the `SetContentsViewType` property to `$::DboValue_PSPICE_MODEL_VIEW_TYPE`.
- Deletes the created Capture session

Set the implementation type property to a specific value

```
proc SetImplementationType { } {
set lLibName {<Capture library path>}      #Add path to the Capture library (.olb), such as
D:\test.olb
set lSession [DboTclHelper_sCreateSession]
set lStatus [DboState]
set lNullObj NULL
set pLibName [DboTclHelper_sMakeCString $lLibName]
set lLib [$lSession GetLib $pLibName $lStatus]
set lPkg NULL
if { $lLib != $lNullObj } {
    set lPackageName [DboTclHelper_sMakeCString {test} 0]
    set lPkg [$lLib GetPackage $lPackageName $lStatus]
    if {$lPkg!=$lNullObj} {
        puts "Package Found"
        # Get the device
        set lDevice [$lPkg GetDevice 0 $lStatus]
        set lCell [$lDevice GetCell $lStatus]
        set lNormalPart [$lCell FindPart 0 $lStatus]
        $lNormalPart SetContentsViewType $::DboValue_PSPICE_MODEL_VIEW_TYPE
        $lLib SavePackageAll $lPkg
        $lSession SaveLib $lLib
        }
    $lSession RemoveLib $lLib
    DboTclHelper_sDeleteSession $lSession
    DboTclHelper_sDeleteCString $lPackageName
    }
}
```

Following are some of the other values of `SetContentsViewType` property that can used to set the Implementation Type:

- $::DboValue_NOVIEW_VIEW_TYPE
- $::DboValue_SCHEMATIC_VIEW_TYPE
- $::DboValue_VHDL_VIEW_TYPE
- $::DboValue_EDIF_VIEW_TYPE
- $::DboValue_PROJECT_VIEW_TYPE
- $::DboValue_PSPICE_STIMULUS_VIEW_TYPE
- $::DboValue_VERILOG_VIEW_TYPE
- $::DboValue_BUNDLE_VIEW_TYPE

Displaying Selected Objects Name and Location

The script displays the name and location of selected objects in the Capture command window. To run the script, select objects on a schematic page and source the following script in the command window. You can source the TCL file using the following syntax in Capture command window: `source {TCL File Absolute Path}`.

dispnameandloc.tcl

```
set selectedobjs [GetSelectedObjects]        ###Select objects
set objlength [llength $selectedobjs]        ###Gets the selected objects list length
set state [DboState]
set partName [DboTclHelper_sMakeCString]
for {set i 0} {$i < $objlength} {incr i} {
 set obj [lindex $selectedobjs $i]
 $obj GetName $partName
 set locationx [DboTclHelper_sGetCPointX [$obj GetLocation $state]]
 set locationy [DboTclHelper_sGetCPointY [$obj GetLocation $state]]
 puts [concat "Name:" [DboTclHelper_sGetConstCharPtr $partName] "LocationX:" $locationx
"LocationY:" $locationy]
}
DboTclHelper_sReleaseAllCreatedPtrs
```

Modifying Selected Wire Color and Width

The script changes width and color of the selected wire to wide width and RGB(0,128,255) respectively. The COLOR5 has RGB value (0,128,255).

To run the script in Capture, do the following in the command window:

1. Source the script, such as `source {D:\colorWire.tcl}`.
2. Select the wire object and run the following command: `ChangeWireColorWidth`

⚠ You can change the width and color of the selected wire to different values.

colorWire.tcl

```
#info vars *DboValue_COLOR*
#info vars *DboValue_*_WIDTH*
proc ChangeWireColorWidth { } {
 set selectedobjs [GetSelectedObjects]
 foreach wireobj $selectedobjs {
  $wireobj SetColor $::DboValue_COLOR5
  $wireobj SetLineWidth $::DboValue_WIDE_WIDTH
 }
}
```

Adding Property to a Selected Wire

The following script adds a property to the selected wire object in Capture. The property added in this script has `PHYSICAL_CONSTRAINT_SET` as property name and `50_ohm` as value. The script also updates the width and color of the selected wire object to medium width and the COLOR8 color.

To run the script in the command window, do the following:

1. Source the script in command window, such as `source {D:\addWireProp.tcl}`
2. Select the wire object and run the `AddWireProp` command.

addWirePop.tcl

```
proc AddWireProp { } {
 set selectedobjs [GetSelectedObjects]
    set lNullObj NULL
    set schOcc [GetInstanceOccurrence]


 set wireProp [DboTclHelper_sMakeCString {PHYSICAL_CONSTRAINT_SET}]
 set wirePropValue [DboTclHelper_sMakeCString {50_OHM}]
 set lStatus [DboState]
 foreach wireobj $selectedobjs {
  $wireobj SetColor $::DboValue_COLOR8
  $wireobj SetLineWidth $::DboValue_MEDIUM_WIDTH

  set lNet [$wireobj GetNet $lStatus]
  if {$lNet != $lNullObj} {
      set lSchNet [$lNet GetSchematicNet]
      set lSchNetOcc [$schOcc GetNetOccurrence $lSchNet $lStatus]
   if {$lSchNetOcc != $lNullObj} {
    $wireobj DeleteUserPropValue $wireProp
    $wireobj NewUserProp $wireProp $wirePropValue $lStatus
   }

  }


 }
}
```

For more information about the colors supported by Capture, see Common Tips and Tricks.

2

Common Tips and Tricks

This chapter lists some of the common tips and tricks that make it easy for the user to create TCL scripts for some of the key Capture features. Following are some of the common tips and tricks:

- DRC markers are placed on instance rather than on occurrence. If you place a DRC marker on an occurrence, it will be visible on each and every occurrence of an instance.
- To get the version information of the installed Capture CIS, use the `GetProductVersion` command on the command window.
- To place a particular part from a particular library on a schematic page, use the following TCL command in the command window: `PlacePart <X-axis> <Y-Axis> <Path To Library> <Package Name> <Designator> FALSE`

  For example, if "Gate.olb" is name of the library and "74LS04" is package name, you can use the above script as `PlacePart 5.00 5.00 "D:/Cadence/SPB_17.2/tools/capture/library/Gate.`OLB" "74LS04" `"A" FALSE.`
- To set the properties of a selected part on a schematic page in Capture, use the following TCL command: `SetProperty {Property Name} {Property Value}.`

  For example, if you want to set the Part Reference property on a selected part, use the following command: `SetProperty {Part Reference} {U-1113A}`
- To place a wire on a schematic page in Capture by providing (X1, Y1) (X2, Y2) locations, use the following TCL command: `PlaceWire <X1 Y1> <X2 Y2>`

  For example, if X1 and Y1 are 88.90 and 106.68 respectively. X2 and Y2 are 165.10 and 106.68 respectively. Use the following command: `PlaceWire 88.90 106.68 165.10 106.68`
- To create a new schematic folder in Capture, do the following in the command window:

  ```
  SelectPMItem "./new_proj.dsn"   #In this case, the design file name is new_proj.dsn
  NSchematic SCHEMATIC2   #In this case, the new schematic folder is schematic2
  ```
- To add a new schematic page in a schematic folder, do the following in the command window:

```
SelectPMItem "SCHEMATIC2"  #In this case, the schematic folder name is SCHEMATIC2
NPage SCHEMATIC2 PAGE1   #In this case, the new schematic page is PAGE1
```

- To generate Capture CIS Bill of Material (BOM), do the following in the command window:

```
Menu "Reports::CIS Bill of Materials::Standard" | DialogBox  "OK" “C:/temp/1.xml”
MenuCommand "57927"  | MessageBox "YES" "Save changes to D:/CAPTURE_AUT" | FileDialog  "OK"
D:/TEST.BOM"
#Update the path (D:/TEST.BOM) as per the requirement
```

- Following are the common colors types that are supported by Capture using a `ColorT` Enum:

```
COLOR1= RGB(255,255,128) //
COLOR2= RGB(128,255,128) //
COLOR3= RGB( 0,255,128) //
COLOR4= RGB(128,255,255) //
COLOR5= RGB( 0,128,255) //
COLOR6= RGB(255,128,192) //
COLOR7= RGB(255,128,255) //
COLOR8= RGB(255, 0, 0) // RED
COLOR9= RGB(255,255, 0) //
COLOR10= RGB(128,255, 0) //
COLOR11= RGB( 0,255, 64) //
COLOR12= RGB( 0,255,255) //
COLOR13= RGB( 0,128,192) //
COLOR14= RGB(128,128,192) //
COLOR15= RGB(255, 0,255) //
COLOR16= RGB(128, 64, 64) //
COLOR17= RGB(255,128, 64) //
COLOR18= RGB( 0,255, 0) // #####GREEN
COLOR19= RGB( 0,128,128) //
COLOR20= RGB( 0,64,128) //
COLOR21= RGB(128,128,255) //
COLOR22= RGB(128, 0, 64) //
COLOR23= RGB(255, 0,128) //
COLOR24= RGB(128, 0, 0) //
COLOR25= RGB(255,128, 0) //
COLOR26= RGB( 0,128, 0) //
COLOR27= RGB( 0,128, 64) //
COLOR28= RGB( 0, 0,255) // #####BLUE
COLOR29= RGB( 0, 0,160) //
COLOR30= RGB(128, 0,128) //
COLOR31= RGB(128, 0,255) //
COLOR32= RGB( 64, 0, 0) //
COLOR33= RGB(128, 64, 0) //
COLOR34= RGB( 0, 64, 0) //
COLOR35= RGB( 0, 64, 64) //
COLOR36= RGB( 0, 0,128) //
COLOR37= RGB( 0, 0, 64) //
COLOR38= RGB( 64, 0, 64) //
COLOR39= RGB( 64, 0,128) //
COLOR40= RGB( 0, 0, 0) // #####BLACK
COLOR41= RGB(128,128, 0) //
COLOR42= RGB(128,128,64) //
COLOR43= RGB(128,128,128) //
COLOR44= RGB( 64,128,128) //
COLOR45= RGB(192,192,192) //
COLOR46= RGB( 64, 0, 64) //
COLOR47= RGB(255,255,255) // #####WHITE
```

- To add a library in Place Part dialog box, use the following command in the command window: `PlacePartAddLib <library absolute path>`
  For example if the library is **Gate.olb**, you can use the following command to add **Gate.olb** in

the Place Part dialog box: `PlacePartAddLib {D:/Cadence/SPB_17.2/tools/capture/library/Gate.olb}`

- To remove a selected library from the Place Part dialog box, use the following command in the command window: `PlacePartRemoveLib`
  This command will remove any number of selected libraries from the Place Part dialog box.
- You can lock a schematic page, unlock a schematic page, and get lock status of the schematic page using the following TCL commands:

lockActivePage.tcl

```
###The following TCL command locks the active schematic page
set lPage [GetActivePage]
$lPage SetBitmask $::DboBaseObject_MASK_LOCKED
$lPage SetBitmask $::DboBaseObject_MASK_TEMPLOCK
```

unlockActivePage.tcl

```
###The following TCL command unlocks the active schematic page
set lPage [GetActivePage]
$lPage UnsetBitmask $::DboBaseObject_MASK_LOCKED
$lPage UnsetBitmask $::DboBaseObject_MASK_TEMPLOCK
```

dispLockStatus.tcl

```
###The following TCL command displays the lock status
set lPage [GetActivePage]
set lLockStatus [$lPage IsObjLocked]
```

- To get the pin type of a selected pin, use the following TCL commands in the command window:

getPinType.tcl

```
set lstatus [DboState]
set pinobj [GetSelectedObjects]
$pinobj GetPinType $lstatus
```

The script will return an integer value. To interpret the returned integer value, see the following table:

Integer	Pin Type
0	DboValue_DBO_IN
1	DboValue_IO
2	DboValue_DBO_OUT
3	DboValue_OC
4	DboValue_PAS
5	DboValue_HIZ
6	DboValue_OE
7	DboValue_POWER