

University of California, Santa Cruz
ELECTRICAL ENGINEERING DEPARTMENT

PCB EDA Tutorial
Laboratory 1, Capture (PSB 17.2)

EE174, Intro. to EDA Tools
 S.C. Petersen

1. INTRODUCTION AND OBJECTIVES

I assume you have read the short essay, *The PCB EDA Toolchain*. It introduces background concepts necessary to intelligibly understand this first laboratory discussing front-end board-level project organization dealing specifically with creating schematics. The relevant EDA tool we will be using here is *Cadence Allegro Design Entry CIS*, or just hereinafter referred to as “**Capture**”. The next lab continues on with an introduction to the printed circuit board (PCB) layout phase, where you will design and generate the so-called “artwork” necessary to make real PCB’s. You should have a printed copy of the reference schematic from Appendix-A available to refer to as you work through this lab. Throughout these labs, “capture or capturing” will refer to the generic procedures associated with expressing and distilling electronic designs in graphical form, whereas “Capture” will always refer specifically to *Cadence Allegro Design Entry CIS*.

Capture has a sophisticated parts creation and management feature, “Capture CIS”, but is otherwise relatively poor respecting rudimentary things we routinely expect from real mechanical drawing programs (CAD tools), like full freedom to create arbitrary shapes, scale them, set font locations and sizes, etc. Hence, although they needn’t be, many of the native symbols are poorly drawn because engineers typically seem to tend to minimize this aspect of engineering documentation. Where possible, I have corrected these deficiencies by providing you with better drawn capacitors, resistors etc. With a little attention to detail, many of the graphic annoyances can be ameliorated. Since acquiring Capture from Orcad, Cadence has corrected some of these deficiencies, enabling designers to produce better-looking and higher quality engineering schematics. The tool’s major strengths, however, lie with the fact that it has a sophisticated and well organized graphical user interface with necessary sub-tools nicely integrated into the program. These are the Component Information System, or “CIS” features which we won’t cover in this course. You will definitely want to explore how to use CIS in the future, especially if you work on collaborative projects having a common source of library parts. Many companies start off junior engineers by having them learn the inventory by maintaining their proprietary CIS projects database. This is an important topic, but beyond the scope of what we want to accomplish in this course.

2. TUTORIAL PROJECT

The tutorial project involves two printed circuit boards that together make a hand-held remote control for transmitting telemetry commands to an autonomous robot originally designed by me for a class project during the Summer of 2005. It was built and tested by students at that time. The heart of the electronic design is a PLA to interpret various pushed buttons to control a state machine that in turn generates specific serial data transmitted over a wireless link. You will capture and layout the handset PCB having the remote control buttons, PLD, oscillator and encoder chips, power supply and misc. capacitors, resistors and LED’s. This board also will have two single row 10 Pin headers (also known as “stake headers”) that provide a socket for plugging in a 902-928MHz RF transmitter as a stand-alone daughterboard. The RF board has already been designed and layed out for you (by a previous student in EE123A/B) and is included for completeness and reference purposes.

3. BEFORE YOU BEGIN . . .

Always plan on organizing your work before beginning by first creating a set of working project directories in your student account. When you first log onto any lab computer, a local profile will be created for you on that particular machine. Since files are local to particular machines, plan on having a thumb drive (or cloud account) to both backup your entire folder tree and provide a means to move it between profiles on different computers. Hence, you will always be working with a folder tree on any local machine having the same specific path: **c:\Users\[your personal login account name]** . Somewhere under this you will create the following set of sub-directories:

...Tutorial	- Top-level directory.
Appnotes	- Relevant application notes, papers and references.
Components	- Mechanical footprint drawings, vender part references, bom's etc.
Datasheets	- Technical datasheets for all devices used in this project.
Doc	- Document files
Handset	- Tutorial project folder containing schematics and pcb layout files.
backup	- Autobackup folder.
pcb	- Board-level PCB files you will be designing.
artwork	- Exportable Gerber and drill files.
dumplib	- Exported footprints and padstacks
logs	- Stuff Allegro produces to track everything it does
netlist	- Netlist output files from Capture
plots	- printing plot files
reports	- More stuff Allegro produces.
signoise.run	- More stuff Allegro produces for advanced analysis
Lib	- Library files & related folders.
Local	- Configuration files for Allegro PCB.
parameter	- Parameter files (*.prm).
tech	- Technology constraint files (*.tcf).
templates	- Empty "template" board files (*.brd).
modules	- PCB layout files stored as complete modules (*.mdd).
padstacks	- Local PCB surface-mount (SM) pads and through-hole definitions (padstacks: *.pad); used with footprints.
symbols	- Local PCB footprints: (*.dra) and various compiled symbol files (*.sm).

Go to our class website, select menu link: *Tutorial Folders*, and download one of the compressed files, *EE174 Tutorial Project Tree* as a Zip or RAR file). Expand it somewhere in your profile's account (note that some of these directories may be empty). *Appnotes*, *Datasheets* and *Components* should synchronize with material that may also be placed in your engineering notebook¹ under these same headings. Generally, these particular directories serve as source repositories for information you find important enough to then print. I usually put many files in them gleaned from web sources, but only print what I actually use and use the remainder for reference. Note that the "padstacks" and "symbols" folders contain files typically unique to your project that otherwise supplement those already available in the native Cadence folders with the same names. I have put in them files used with this class.

Since many students later use this same folder tree when beginning new projects in later courses, I have provided a single compressed file (*default project tree* as a Zip or RAR) that can be conveniently expanded to automatically create a ready-to-go set of folders. After expansion, several folder names will be shown in brackets indicating that you should rename them according to your particular project. Note too that the padstacks and symbols folders will be empty.

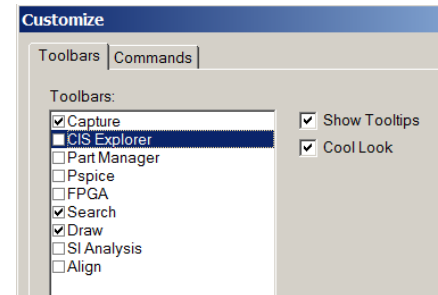
During the process of using Capture and Allegro, especially Allegro, additional spurious directories may be automatically created. Don't be too concerned about these or their contents since they typically contain information related to advanced simulation or auto-routing features that we won't be concerned with in this course; just consider them "more stuff Cadence produces!"

¹ See *The General Practice of Engineering Notes*, rev 3.5, found on our website.

Note: as you work through these labs, many of the specific directory paths shown in the screen shots will be different than yours. This is simply a consequence of working on a different machine while writing the tutorials; disregard them! I will, however, often use ellipsis' ("...\") to denote absolute directory structures having some specific but irrelevant machine-dependent root path.

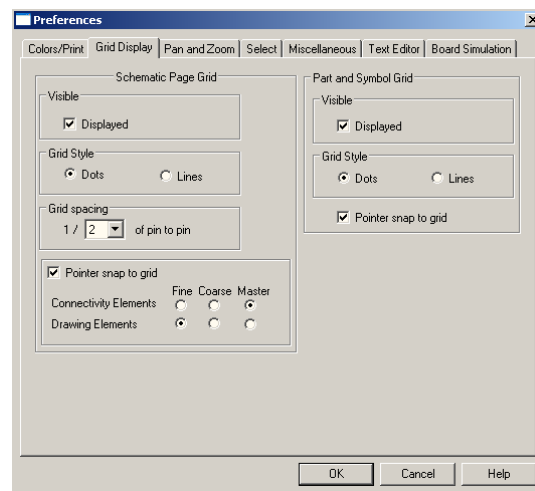
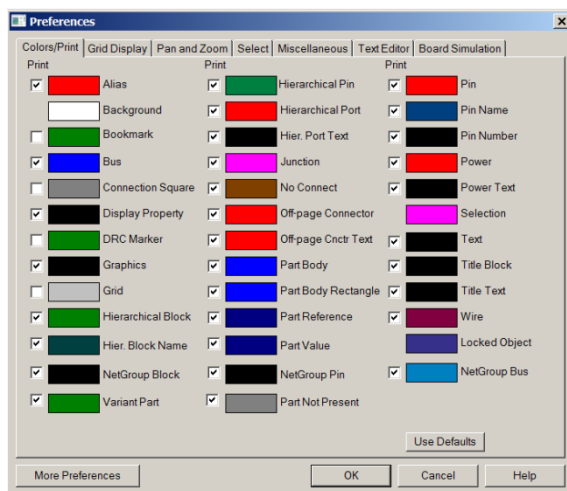
3.1 SETTING GLOBAL PREFERENCES

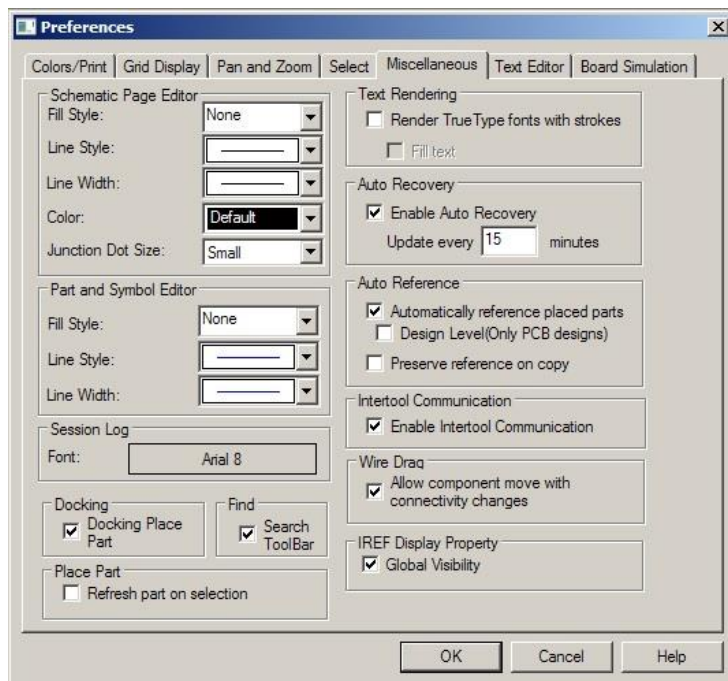
First, boot Capture by selecting Allegro Products -> Capture CIS from the main menu and under Cadence Product Choices choose *Allegro PCB Design CIS L* from among the three choices presented. Begin by customizing the toolbars as shown to the right by following menu sequence **Tools** → **Customize...** We won't be using PSpice, FPGA, CIS Explorer or PartManager, so leaving these unchecked unclutters the desktop. These settings apply to anyone using the lab computer you are now on, so please don't modify them, except for Tooltips and Cool Look. If the default toolbars do perchance get changed, this dialog can be accessed to reset them. You may want to uncheck Cool Look if you prefer spatially delimited icons. Similarly, Tooltips are useful until various objects become very familiar and may become a nuisance thereafter.



Before creating schematics or part symbols for the first time, you should also setup a number of global defaults found in the *Preferences* and *Design Template* dialogs. Select **Options** → **Preferences...** This will bring up the seven tabbed dialogs devoted to various Preferences.

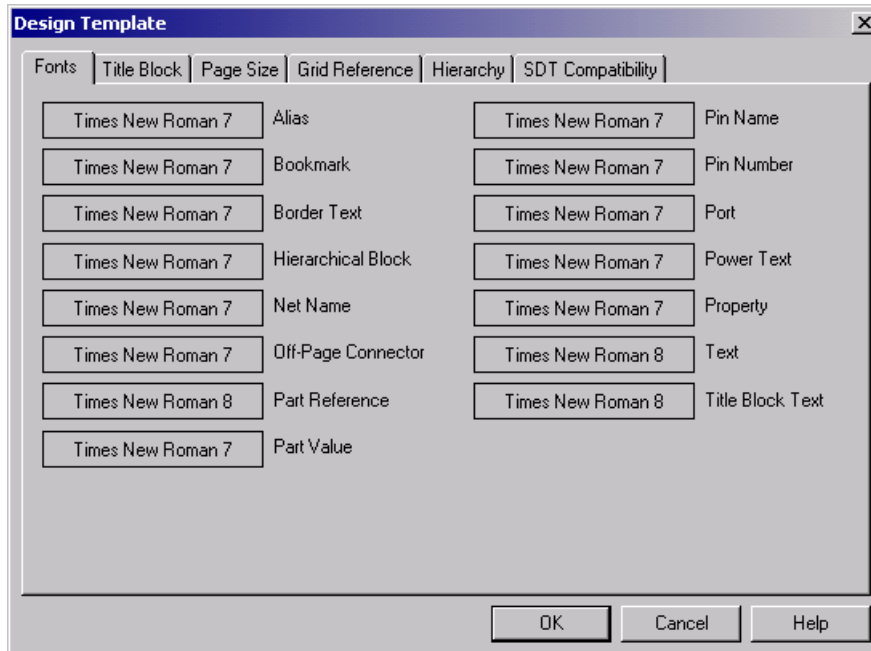
Working with the grid is sometimes a problem in Capture, so it's nice to be able to see it. Find and set the **Grid** object in the Colors/Print tabbed field for a dark, but not too dark, color (simply click the colored rectangle to see the color menu). You'll soon enough be able to see the effect of this and change it if you like. The checkbox options all indicate whether we want associated objects to also appear when printing a hardcopy; leave the checkbox for the grid unchecked (your hardcopies will look very weird if you check it!). Click on the **Grid Display** tab to show the fields allowing you to define how the grid will look when working with schematics or parts symbols. Set these options as shown for both tabbed displays.



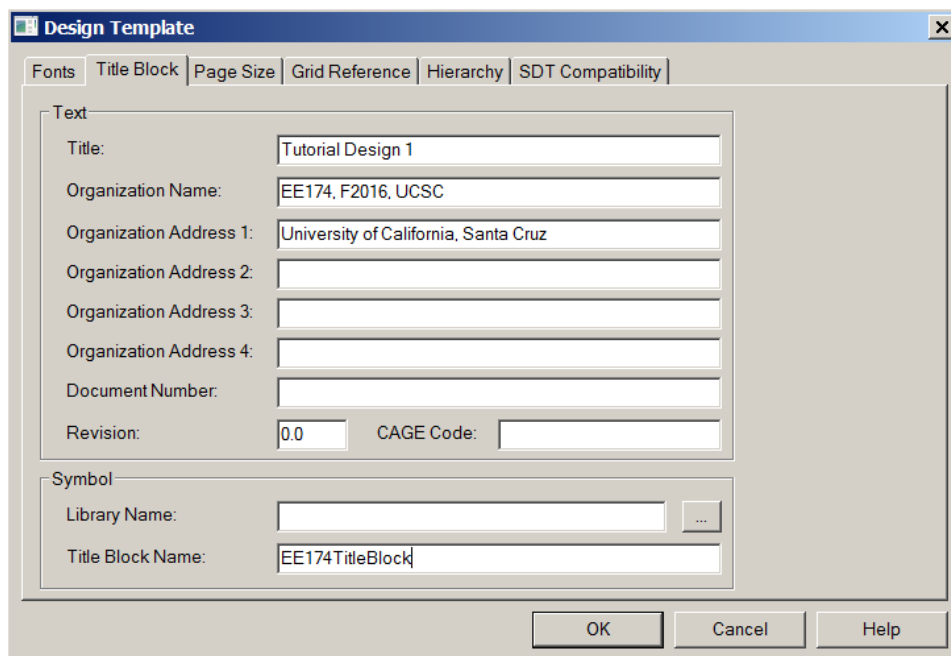


The Miscellaneous tab is important, so be sure it's set correctly. Fields should be as shown until you learn how things work. In particular, the Wire Drag must be checked. Again, since others will be using this lab computer, try not to meddle with these settings, but if something does get changed, you know where to go to fix it. Just use the default values for all objects on the remaining tabs. Close the Preferences dialog and proceed to the Design Template dialog.

Select **Options** → **Design Template...** This will bring up the six tabbed template dialogs. Unlike Preferences above, these various Design Templates don't affect the appearance or behavior of any schematics or pages already in existence, rather they apply only to new schematics or pages added after setting them. Capture does, however, allow some of them to be changed (page size and grid reference) on a per-page basis by selecting menu **Options** → **Schematic Page Properties...**



The **Fonts** tab allows fonts to be set along with their respective sizes according to preference. Unless you know what the sizes look like, leave them alone. I like the Times New Roman fonts over the default Arial fonts and have defined them as shown.

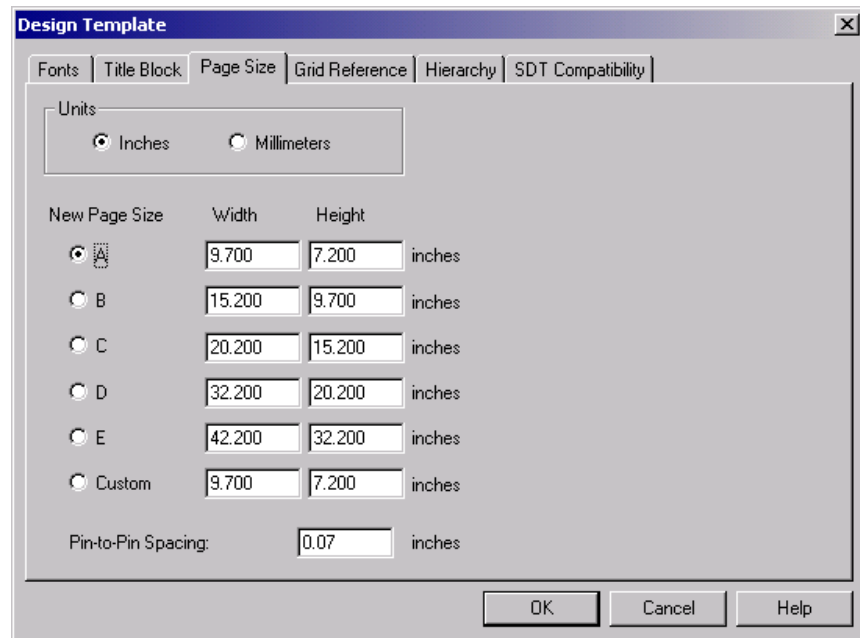


The **Title Block** tab shows the *properties* associated with any of the many *default* title blocks supplied by Capture. Not all of these properties are used in every instance. We will be using a custom title block that uses some of these as display properties, namely the first two and Revision (Org. addr. 1 is not displayed but carried along as a hidden property). Fill them out as shown (change the class and date you're taking this class!).

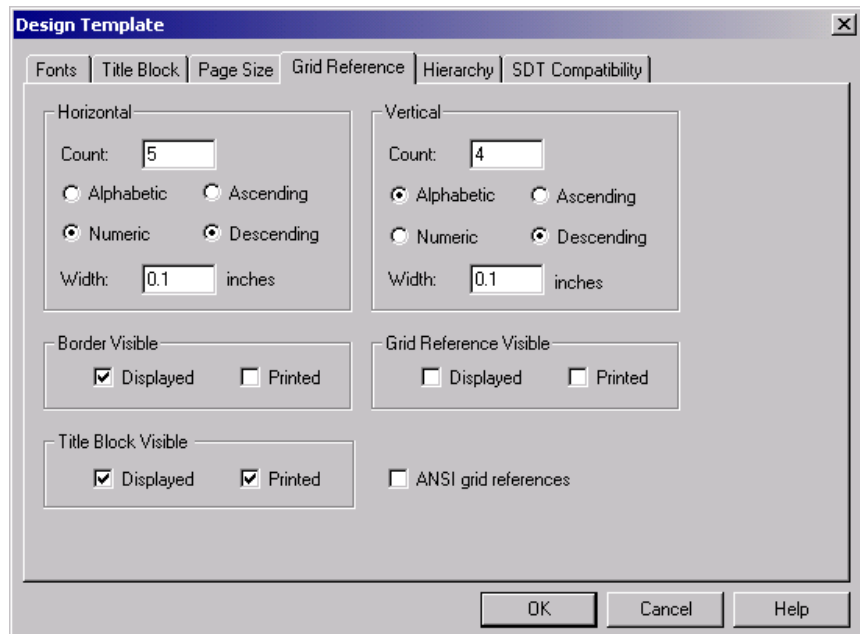
The custom title block, *eepe174titleBlock*, can be found in the EE174REFLIB.olb library that should be in your **Lib** folder created earlier. Since this will certainly be in a different

path than that shown, type in the full path and library name or just navigate to it by using the ... icon.

The **Page Size** properties tab define units, dimensions and scaling. Since we will be working with A-size drawings in this tutorial, scaling everything down by 30% makes a much nicer looking page. Do this by setting the **Pin-to-Pin Spacing** property. The 100% reference spacing is always 0.1 inches, so we can increase the size of the workspace with respect to the symbols by simply decreasing this quantity. I have found a 30% reduction, or spacing of 0.07 inches gives a nice part size to work with. This is the only property that should be changed on this page.

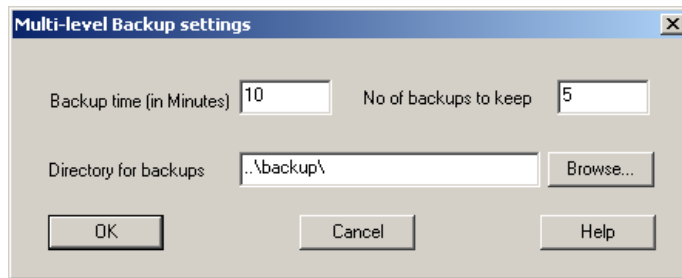


The **Grid Reference** tab defines how the border grid and title block are displayed and printed. The default values initially define the appearance of a mechanical drawing having numbered (or lettered) vertical and horizontal grid indices for visually locating parts on the page. On A-size drawings this is unnecessary, so we will change it. We want the border visible but not printed and the title both displayed and printed.



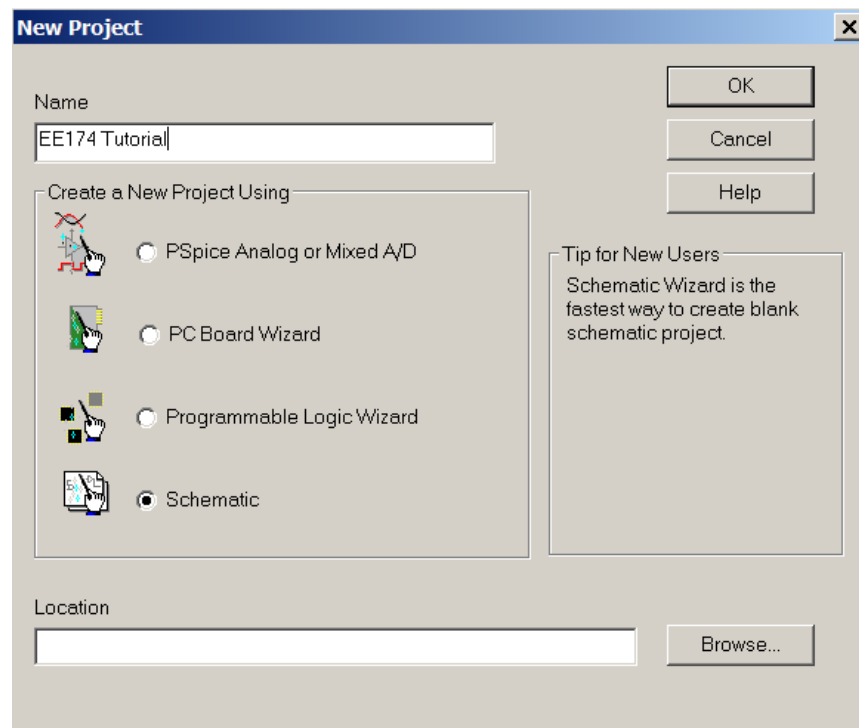
Note: I commented earlier that another menu sequence, **Options** → **Schematic Page Properties...**, available only when particular pages are selected and active, provides a way to set these two dialogs for particular single pages. This is quite useful to know, since creating a new page generally seems to always begin with the *Border Visible Displayed* checkbox unchecked. Note too the pin-to-pin spacing cannot be set except through the Design Template / Page Size dialog above and can only apply globally to all pages in a given design.

Finally, we need to configure Capture to automatically backup our work. Follow menu sequence **Options** → **Autobackup...** to bring up the *Multi-level Backup settings* dialog, and set the fields as shown. This will place up to five new backups of your working schematic file spaced 10 minutes apart in time into your ...Tutorial\Handset\backup folder. Each file in the series will be appended with a number indicating their relative position in the series. Note: this folder must already exist or Capture will complain. So you may have to add it and then navigate to it through the Browse... button.



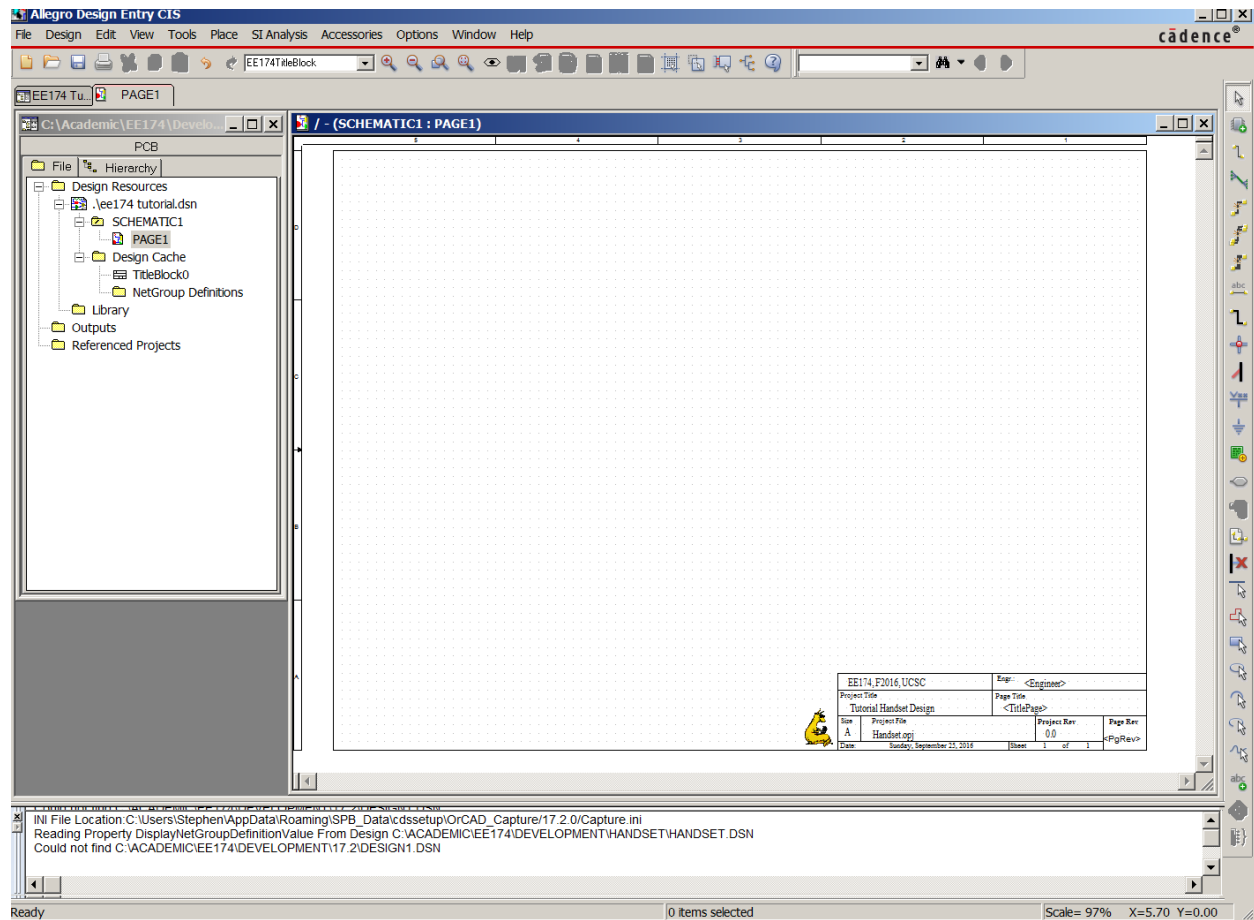
3.2 STARTING A NEW SCHEMATIC PROJECT

Begin creating a new schematic project by selecting the menu sequence **File** → **New** → **Project**. The following dialog box will appear:



For the Location field, Browse to the **Handset** directory in your project folder tree and name the project **EE174Tutorial**. This will create an empty project workspace in the Handset directory and add the project file: EETutorial.opj that looks like the following (after adjusting objects on the screen accordingly).

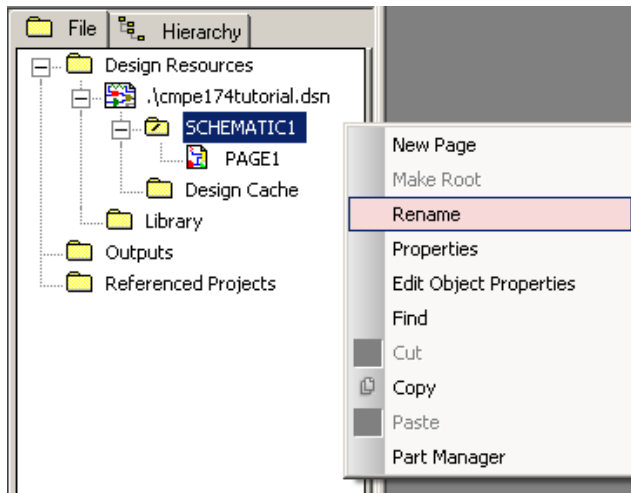
Note: Some of you may wonder why we're not starting with the "PC Board Wizard" instead. After all, the series of captured schematics is aimed at final implementation on a PCB. You can answer this question yourself, but only *after* finishing the entire lab; at that time, go back and create another project using this flow and I think you'll understand why I didn't follow it here (you can, of course, use it in the future if you like).



As initially created, this new workspace contains a default schematic, SCHEMATIC1 containing one page, PAGE1. The vertical dialog box on the far left is called the **Project Manager**, and is one of the strengths of Capture that makes working with complex projects pretty easy. It contains a hierarchy of virtual folders (*i.e.* they don't actually exist as physical directories on the computer) displaying how the project's workspace is logically organized. The horizontal window at the bottom is the **Session Log**. It records just about everything done – useless as well as useful (control-delete erases it). Clicking the “x” in its upper left corner closes it, while following menu sequence **Window** → **1 Session Log** brings it back again. It is most useful when checking designs using the DRC (design rules check) and when compiling the whole project at the end; these topics will be fully discussed later.

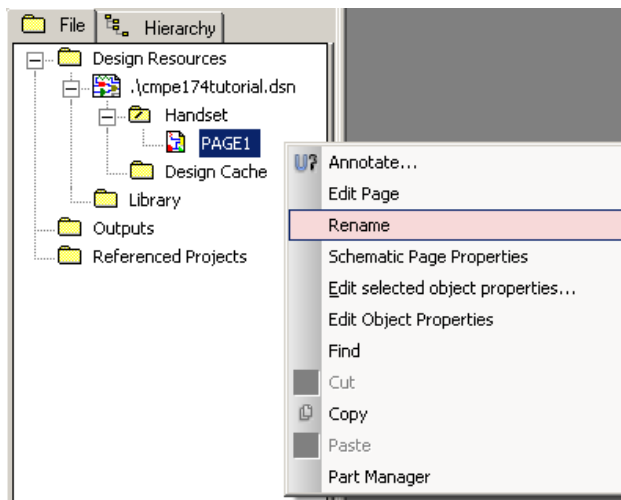
Note the title block shown on PAGE1 and referenced in the Design Cache comes from the EE174REFLIB.olb library file entered earlier on the **Design Template** → **Title Block** dialog (see pg. 4). If it's not present at this point, namely after a page is in existence, it can only be added manually. This is useful to know, but requires a deeper understanding about how Capture manages symbols generally. Hence, if it is missing, don't worry about it now as we'll discuss it later after learning more about symbols.

Adding New Pages - Our project will consist of three pages, one for revision history and two documenting the full engineering design. We will add these pages now. Do this by first changing “SCHEMATIC1” to “Handset”, changing “PAGE1” to “1 Revisions” and adding two new pages titled “2 Power and Oscillator”, and “3 Wireless Transmitter”. Prefixing page names with numbers forces them to appear in the correct order, since Capture displays them alphabetically.

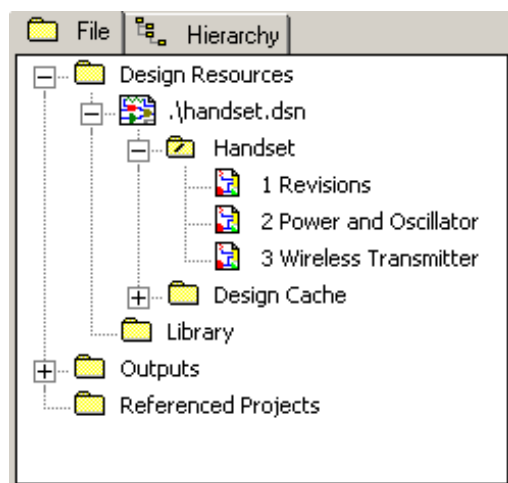


The following sequence shows in detail how this is done:

Change SCHEMATIC1 to “Handset” by right-clicking it and choosing **Rename**. Note that PSB17.2 has a few more options than those shown from an earlier version.



Similarly, rename PAGE1 by right-clicking it and choosing **Rename**.



Add the two new schematic pages by first highlighting Handset, right-clicking and selecting **New Page** each time. You will be asked to confirm the suggested default name or rename them each time. When finished, Handset should contain the three pages as shown.

The Revision History Page - Activate the Revision History page (double click on “**1 Revisions**”) and construct the page with headings using drawing objects from the drawing menu on the far right of the desktop (but don’t add any entries yet). Useful relevant tools are:



Select tool. Use this to select any object for editing.



Line tool.



Place Rectangle

After placing lines or rectangles, (draw it, right-click and select End), they can be edited for weight (thickness) and color by using the Select tool above to choose a particular line or rectangle, right click and activate Properties.



Add Text tool.

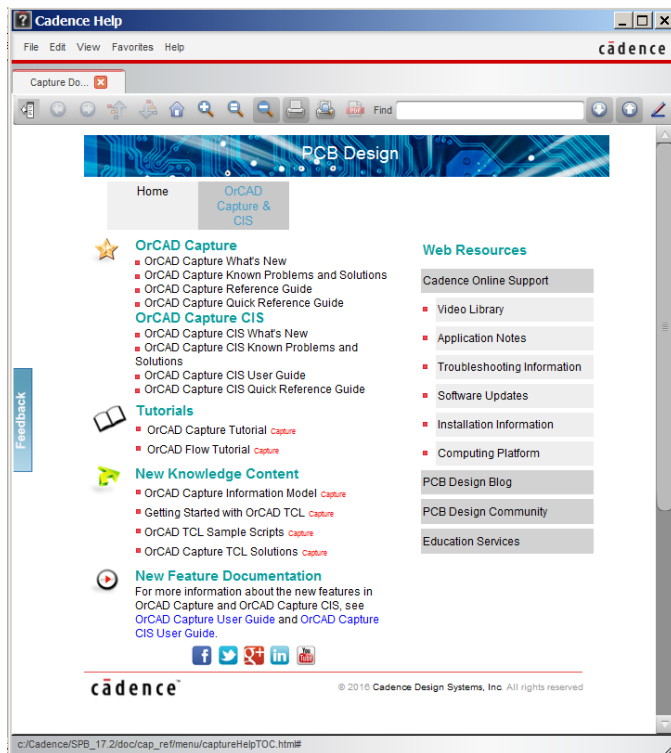
Obviously, this tool is used to place text. It’s very easy and versatile to use.

This page is especially useful when multiple engineers work on a single project. If you’re the only one, its really optional, but still may be useful nonetheless. I’ve included it in this lab to acquaint you with the utility of having one and how to use it effectively. If you don’t use such a page, and I sometimes don’t, entries must be recorded in your engineering notebook to equivalently track changes. Only make entries whenever design changes are made to the real schematic pages that should be noted to avoid later confusion. You will find that most projects don’t fill up the whole page, but if they do, just add another. Entries should briefly explain what each entered version change is about, but avoid copious lengthy discussions; these should be entered in your engineering notebook. As noted, version 0.1 is the initial creation of the project. Go ahead and make this entry now.

Revision Guidelines - Revisions between 0 and 1 document changes made during the first design entry phase leading to a complete prototype. Thus, a suggested sequence might be:

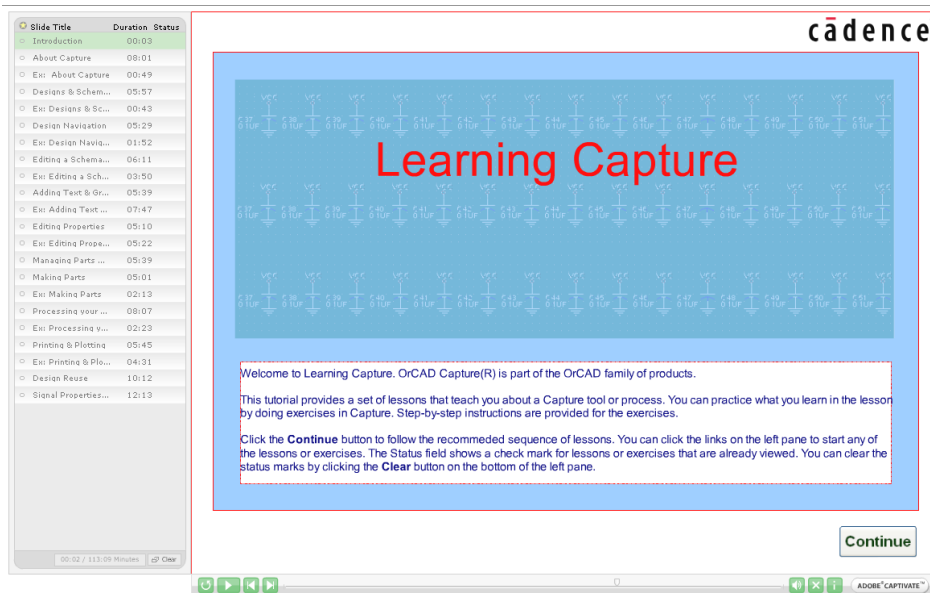
- 0.1 Initial project creation.
- 0.2 Fully entered schematic symbols.
- 0.3 All schematic symbol properties checked against their respective datasheets.
- 0.4 All schematic symbols mapped to checked (against the design *and* datasheets) and correct pcb footprints.
- ...
- 0.9 Successful DRC (“design rule check”) and final netlist creation.
- 1.0 First complete prototype, combined schematics and PCB layout finished and sent to boardhouse.

Ostensibly, version 1.0 results in a real bare printed circuit board to which parts are then added and experimentally tested. Versions between 1 and 2 should reflect modifications since version 1.0 was completed. This typically includes design changes, correction of wiring errors, footprint revisions etc.. I generally also keep printed hardcopies of these schematics in my engineering notes so I can keep a written record as penned comments directly on the pages. How this is done is pretty much up to you and your engineering team. Companies typically have strict policy requirements, but if you’re working as an individual, make it work clearly for you.



Cadence Help Resources - Take a moment to navigate to the Help menu and get a quick overview of the workspace by stepping through the About Capture lesson: menu sequence **Help** → **Learning Orcad Capture**. You should see the screen shown at left. Click on the field under the Tutorial block: *OrCad Capture Tutorial*. This will bring up a browser based (html) interface shown below, titled “Learning Capture”.

Click on the **About Capture** menu item along the left bar and watch presentation for a quick orientation to the project window and its contents. Do the same for **Designs and Schematics**. The ghost of Orcad lives on in these early tutorials, since Cadence hasn't updated most of them much since they first acquired Orcad until version SPB



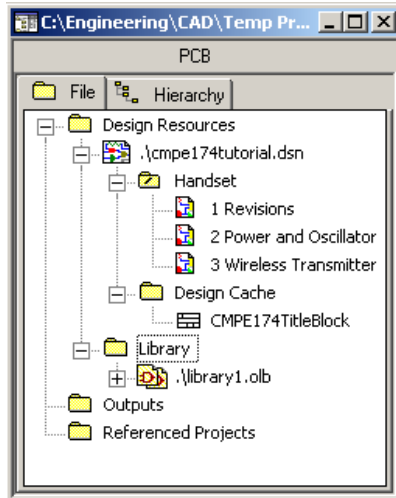
16.5. Here, they animated them and have made the exercises interactive. I suggest the exercises be skipped since they usually aren't worth the effort. The aim here is more to get you familiar with where things are than to learn how to use them. Although this is a relatively poor reference set of “lessons”, it is quite useful as a quick refresher on things you may have later forgotten. Cadence has removed some material from earlier versions and put them elsewhere and added new material

Even though we won't be using the CIS features in Capture, you should read through the description of what it is to get a high-level grasp of essential features by opening the **OrCAD CIS User Guide**. Using the Windows File Manager navigate to **C/Cadence/SPB_17.2/doc/cisug/disugTOC.html** and open the HTML file in a browser. From Chapter 1 select *What is the OrCAD Capture CIS System?* Proceed to read through this document. As you should see, setting up and using a well-maintained CIS database constitutes a complete project in itself, and is best used where large managed part inventories are reused over many projects. After learning board-level design in this class, you will then be in a better position to consider whether CIS is appropriate or not. Note the **...SPB_17.2/doc** folder is where all help references reside. It is sometimes useful to access things you want directly through this folder tree than to wade your way through the otherwise byzantine Cadence access paths.

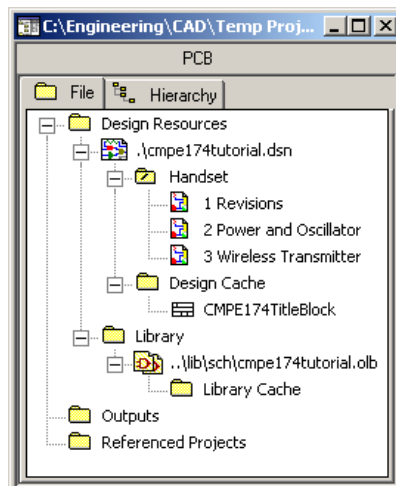
3.3 WORKING WITH SCHEMATIC PARTS' (SYMBOL) LIBRARIES

Besides the many symbol libraries Capture provides us with, we will very often want to create custom symbols of our own for parts that don't already exist, or to modify those that do. These new or modified parts should not be saved in Capture's native libraries, but rather in a custom project library created by you. Capture symbol libraries all have the extension ***.olb** (historically for "Orcad Library")³; we will make use of these later. We will now create several custom symbols for certain parts not already available and save them in a custom library associated with your project.

First, create and add a new library by following menu sequence **File** → **New** → **Library**. Capture will automatically create and display a new empty default **library1.olb** in the virtual library folder, as shown below.



At this point, the library exists only in memory. To save it as a real file, right-click on the library file name and select **Save As ...**. Name the file **EE174Tutorial** and place it in your **Lib** subdirectory; this should be the same place where EE174RefLib.olb is now located. We are now ready to add some parts to it. Remember, any changes will only be to the virtual file (in memory) and must be followed at some point by saving it to disk (if you forget to do this, Capture will nag you at shutdown).



The screen shot shown at left summarizes the changes we've made so far

³ These libraries are always installed by Cadence in ...\\SPB_17.2\\tools\\capture\\library.

MAKING A NEW PART – Refer to the Handset schematic handed out during class (or downloaded from our website: ...notes\Reference_Schematic_v1.1.pdf) and note on page 3 that we must add the ATF16V8BQL PLD and the HT12E Encoder chips, since these aren't in any of Cature's native symbol libraries. Generally, the easiest way to accomplish this is by first finding one that's close or similar and then copying it into your custom library using the usual clip-and-paste or click-and-drag operations. After that it can be edited, or it can also be created directly. I'll show you both ways now.

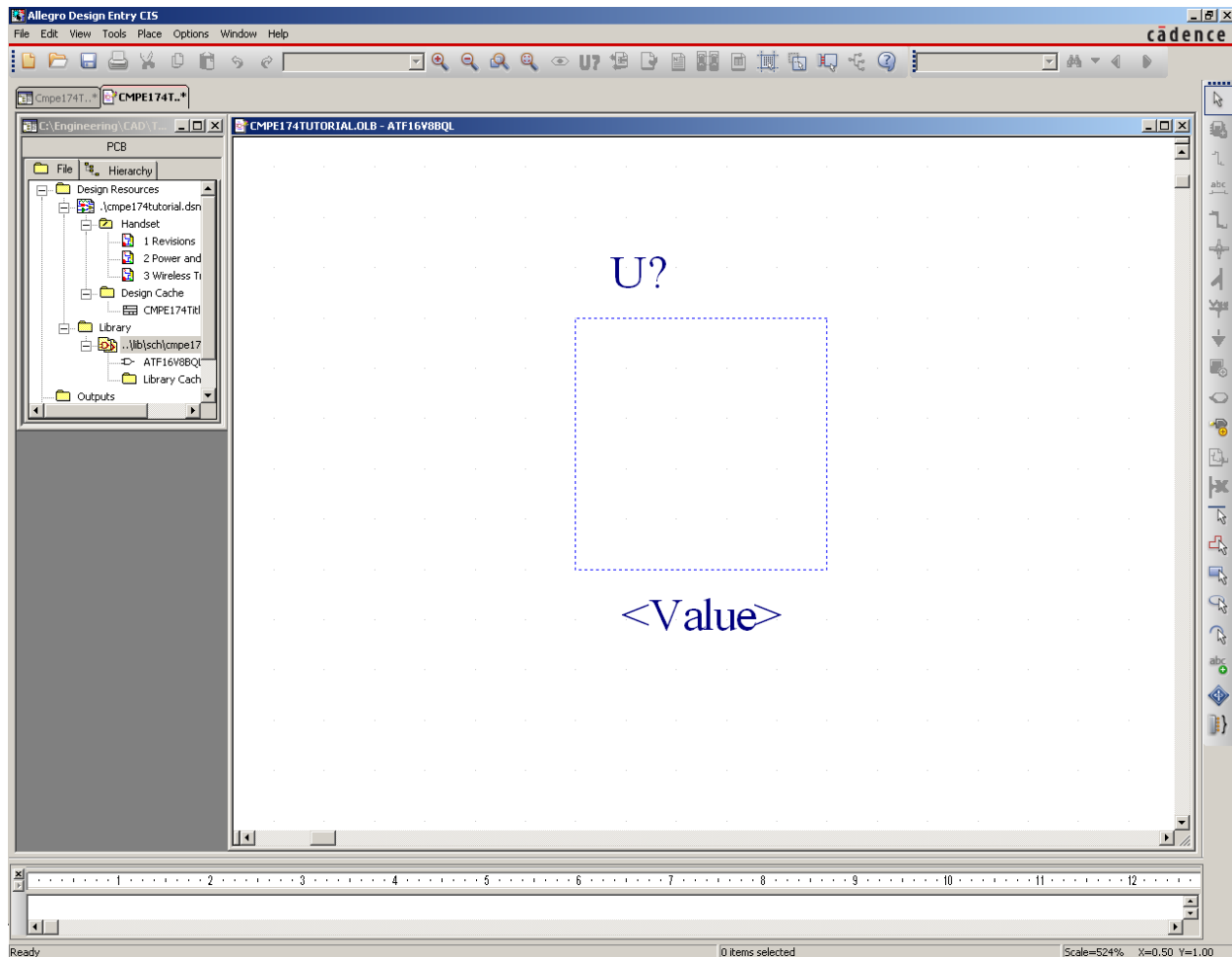
Add a new part to your working project library by first right-clicking the library filename ...lib\ee174tutorial.olb and selecting **New Part** from the pull-down selection list. This will bring up the dialog box for specifying **New Part Properties**, as follows.

Here, we will make a new symbol for the Atmel ATF16V8 PLD. One purpose for having datasheets associated with projects is to specifically resolve design questions. Open the *Tutorial\Datasheets\atf16v8.pdf* and observe this part comes in different packages. The one we'll be using is the 20-pin dual inline (DIP) package. Now we could name the part ATF16V8BQL-DIP to uniquely identify it as a symbol tied to just the 20-pin DIP, but we won't since all the packages have the same pinout (confirm this yourself now by inspection). Rather, we'll just make it a generic part suitable for any footprint and specify the package type later by directly editing the **PCB Footprint** property of the symbol *after it has been added to the schematic*. You'll see how this works at the end of this lab. Leave the **Part Reference Prefix** as U.

Note: Why the reference prefix "U" (these are also known as *Reference Designators*)? Prefixes help us categorize schematic and PCB parts by inspection. A loose informal standard exists for referencing most types of parts. For example, C is used for capacitors, R for resistors, L for inductors, U (meaning "unit") for IC's, X for quartz or ceramic piezo-electric crystals, P for plugs, J for jacks, Q for transistors, SW for switches and so on. Learning and using this informal standard will make your designs more readable and self-documenting⁴. The key idea is *self-documentation*. Producing professional level engineering schematics always strives for this kind of technical transparency, and is considered an informal standard of good engineering practice.

Leave the default values for all the other settings. We'll be discussing these later on. Click **OK** to bring up the workspace for part creation. It should look like the one below. Note that tools for working with the new part can be found on the toolbar at the right-hand side of the screen.

⁴ See *A Note on the Proper Documentation of Engineering Schematics*, pg. 4, found on our website.



Get started by first maximizing the screen to fill the work space, zooming out to make the dashed square smaller, and then drag the dashed line from a corner to make it bigger since you won't get many pins on it otherwise.

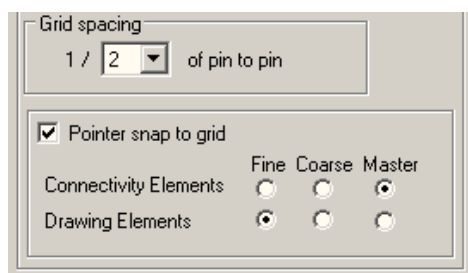
Note! Capture uses grids for both the parts editor and schematic entry. Until you get accustomed to how these grids are used, its best to leave the *snap-to-grid* feature ON while making and editing parts. This is controlled by the **Snap To Grid** toggle icon in the upper toolbar. Mode is indicated by its appearance after selecting with the mouse:



Snap To Grid is ON

Snap To Grid is OFF

TIP: Get in the habit of glancing at the grid icon to mentally keep track of which mode you are currently working in. Failure to do this can cause untold grief when trying to connect pins with wires that don't align properly, but otherwise appear to.

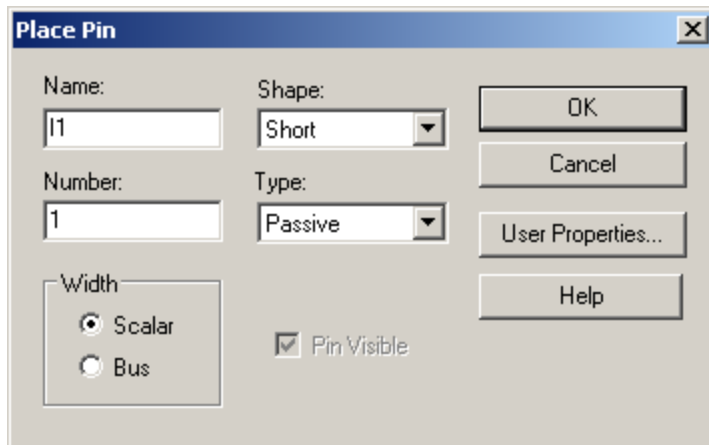



The one exception to this rule is while placing text. Generally, precise placement is only possible when the much too coarse grid is turned off. Hence, it's alright to turn off snap-to-grid while placing text objects. However, pin placement and basic part dimensions should generally be done with grid snap on. After you get some experience, you'll know when to turn it on or off. Fortunately, Cadence recognized this annoying fact in earlier versions and now provides a means to split the grid resolutions so one applies to placing parts and wiring, and the other for text. We took advantage of this feature earlier when we

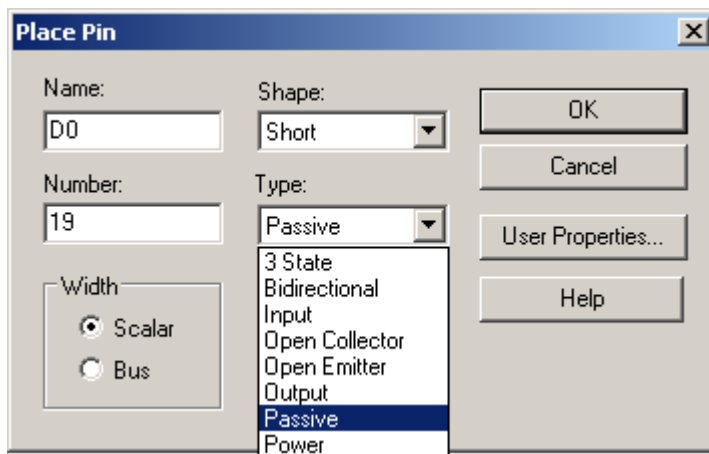
selected the **Options** → **Preferences** → **Grid Display** tabbed dialog box (refer to page 3), shown to the left.

Basically then, Capture has two grids, one course the other fine. The course grid is always based on a pin-to-pin spacing of 0.1 inch, while the fine grid is 1/10 of this. Setting the **Grid spacing** to 2 as shown, makes the course grid 0.05 inches, a setting I prefer. The remaining two lines allow us to define how text (drawing elements, including lines, boxes etc.) resolves and how “Connectivity Elements”, like parts and wires resolve. The settings shown are equivalent to doing what we used to have to do by manually by toggling the Snap to Grid icon noted earlier. Read more about these settings by selecting Help in the lower right corner of the Grid Display dialog.

Continue by referring to the part’s datasheet in your Datasheets folder to confirm the correct pinout for the 20-DIP package. Note that since this component is a PLD (programmable logic device), the pins have been renamed according to their function as I used them in this design.

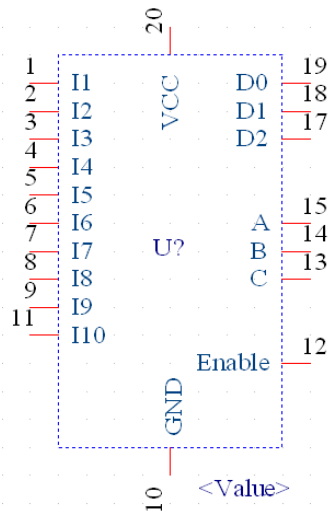


Use the **Place Pin**  tool to bring up the Place Pin dialog box to place pins on the dotted work-space rectangle beginning with the IO pins, I1 through I10. You should experiment with this tool to familiarize yourself with how it works. Pins can be placed sequentially or individually. Each pin must have a “Shape” and “Type” depending respectively on how it should appear and what its’ electrical function is. Here, use the default pin **Shape** (line) or make them short by selecting “Short” from the pull-down list. Note that zero-length is mostly used with invisible power pins which we won’t be using in this tutorial.



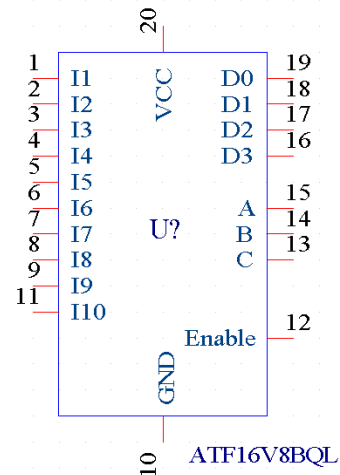
Each pin’s electrical behavior must be properly identified by choosing one of eight types found in the **Type** property pull-down list. This is used by the Design Rule Checker (DRC) to identify improperly wired designs. For example, wiring two pins both defined as Outputs would flag an error, whereas an Output and a Bidirectional pin would flag only a warning. Passive components, like resistors, would have pins identified as Passive. Thus, Pins I1 through I10 are inputs, D0 through D3, and A, B, C and Enable are all outputs. VCC and GND are power pins. These latter pins are often made invisible. Unless you have considerable experience and understand how to

use invisible pins, they should be left visible, so be sure the **Pin Visible** box is checked for VCC and GND. This is usually good practice anyway, since we can then explicitly indicate, for example, bypass capacitors associated by design for these pins directly on the schematic. Digital designs often have these pins invisible since power and ground are typically routed directly to inner power planes. Analog designs should always show them since filtering and other circuits are often part of the power design.



the part, ATF16V8BQL.

The finished part with solid outline and defined name should finally look like this:

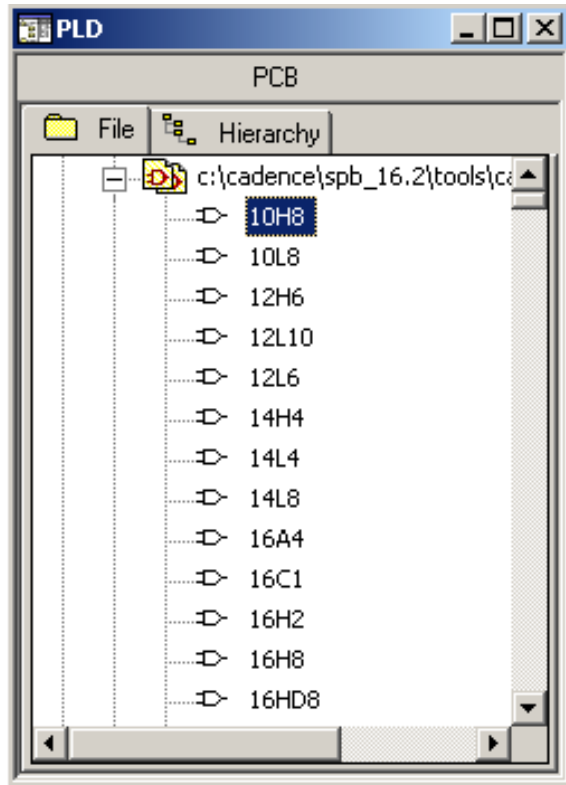


Save your new part and close the window. It should now be visible in your custom library, ready for placement on the schematic.

Generally try to adhere to the time-honored schematic standard of *placing inputs on the left and outputs on the right* whenever possible. This applies to schematics as well as part symbols. Similarly, power pins should be arranged so higher potentials are located near the top and grounds near the bottom of the part (or schematic). These are guidelines, and can always be broken for good reason (and we often have them). Note too that *we are not* creating a facsimile of the actual part as it would physically appear, but a *functional block diagram* that puts a higher priority on any pin's function rather than its actual physical location. Hence, notice that I placed the pins functionally rather than the way they would otherwise appear on the actual 20-pin DIP package. Oddly, beginners always seem to want to draw these symbols physically rather than functionally; understand the differences now and seek to create functional symbols rather than physical symbols. Later we will map these to real physical footprints for the PCB phase.

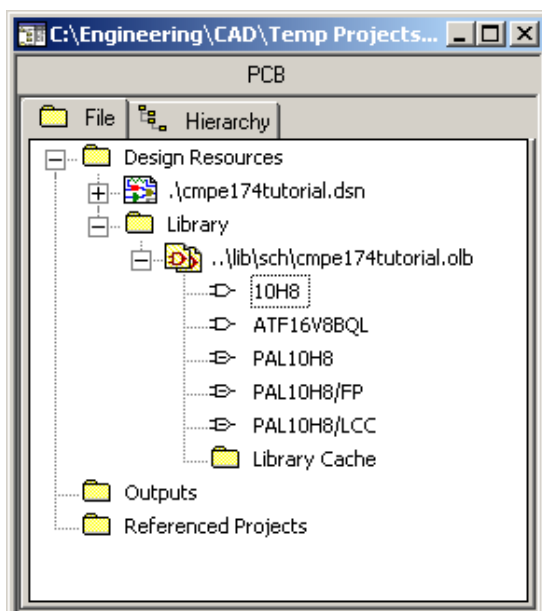
After finishing the pins, choose the **Place Rectangle**  tool and add a solid line overlaying the dotted-line workspace. Double-click on the <Value> field and name

USING AN EXISTING PART - First, find a suitable part in Capture's libraries by menu sequence **File** → **Open** → **Library**. Notice that Cadence keeps all of the Capture library files in the path: ...tools\Capture\library. Find the library called **PLD.OLB**. This will open up a new window listing the library's contents. From here, any of the parts are viewable by double-clicking them. Since the AFT16V8BQL is a 20-pin DIP, we note that the very first part, 10H8, is pretty close to what we want. So we'll copy that to our custom library. Do this by copying it to the clipboard and then pasting into your library: select the part, copy it to the clipboard with CTL-C (or equivalently use the menu sequence **Edit** → **Copy**), then highlight (navigate to) your library and hit CTL-V (or equivalently use the menu sequence **Edit** → **Paste**) to paste the part into your library.



Important Note: These library files contain thousands of pre-defined schematic symbols. Many of them are poorly or awkwardly created, and some have errors. Unless you intend to edit them or use them as the basis for a new part, always be sure to proof-check them, especially for proper pin numbers and type functions against actual datasheets. Datasheets can typically be found on the web as PDF files and then saved in your Datasheets folder. I have provided a number of these that should already have been downloaded from the Datasheets tutorial folder on our website.

Generic parts are also typically available from many manufacturers, and the quality, completeness and accuracy of these documents can vary considerably. Consequently, some are more useful than others, so rather than downloading the first one you might find on the web, review several of them and save only the most useful for your purposes.



After adding the 10H8 symbol to your library, some extra parts will also magically show up. These are *part aliases* (i.e. the same symbolic part but with different names) and will always be designated with a line inside the little gate icon; these are not needed so just delete them to avoid confusion. In this example, the aliases are the PAL10H8, PAL10H8/FP and PAL10H8/LCC. These have different physical packages but otherwise the same graphic template symbol (the 10H8). So if you wanted, you could create an alias of your ATF16V8BQL that had a different footprint (it would have to be named differently of course). Deleting any alias directly by selecting it will also delete the parent and *all* other aliases. They can only be removed by first opening the parent part (double-click on the 10H8) for editing. Then follow **Options** → **Package Properites** to bring up the *Edit Part Properties* dialog. Select *Part Aliases...* and delete the aliases.

At this point, though, you already have a part created for the ATF16V8BQL, so just delete the newly copied part.

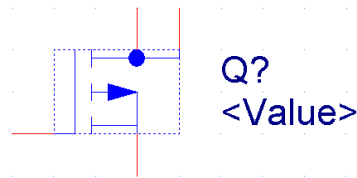
CREATING HOMOGENEOUS PARTS - The two PMOS field-effect transistors, Q1-1 and Q1-2 are examples of a device (the TPS1120) having more than one set of identical *logical* parts inside its' physical package. Capture's help documentation tells us the following about multiple-part packages⁵:

"Parts usually correspond to physical objects---gates, chips, connectors, and so on---that come in packages of one or more parts. Think of these packages as physical parts and the parts you place on a schematic page as **logical parts**. Physical parts that comprise more than one logical part are sometimes referred to as multiple-part packages. For simplicity, Capture usually refers to both as parts.

Logical parts in a package may have different pin assignments, graphics, and user properties. If all the logical parts in a package are identical except for the pin names and numbers, the package is **homogeneous**. If the logical parts in a package have different graphics, numbers of pins, or properties, the package is **heterogeneous**. For example, a hex inverter is homogeneous: the six inverters are identical, except for their pin numbers. A relay, which has a normally opened switch, a normally closed switch, and a coil, is heterogeneous: the three physical parts differ in graphics, number of pins, and properties."

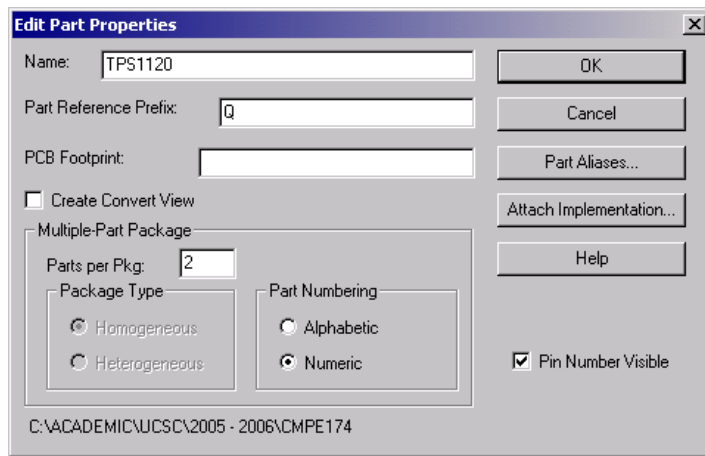
Since the TPS1120 is not in any of Capture's native libraries, we will have to create it. We could just begin and make it from scratch, like the ATF16V8 earlier, but the symbol is most likely already in Capture's native libraries somewhere, so we'll try and find something close and begin with that. First, consult the datasheet for the TPS1120 to get the actual proper schematic symbol, pinout and PCB package (we'll need that later with Allegro). A PDF version of this file should already be in your Datasheets folder. Open it and print relevant pages showing the schematic, pinout and PCB footprint. Please avoid printing all eight pages unless you can print double-sided and specify two pages per side. At this point, most of these pages are irrelevant anyway if all we're really interested in is the symbol, pinout and later footprint. Hole punch whatever pages you do print and file them under the Datasheets section in your engineering notebook binder. Be sure to put a **date, topic, page** and **your initials** in the upper right corner of the first page, and make an appropriate index entry. You can look at one of my (prof. Petersen's) notebooks as an example.

The first page of the TPS1120 datasheets describes it to be a set of PMOS enhancement mode FETs. Rather than having to create it from scratch (dread), peruse Capture's existing libraries for a suitable PMOS FET that can be used as the basis for your new part. After some investigation, we happily find that Capture provides a series of generic symbols in the **Transistor.olb** library. Open it and find the sequence of parts labeled: MOSFET_EP_.... Reviewing these sequentially reveals **MOSFET_EP_GDSD** to be the part we want (for those interested, the cryptic notation stands for: Mosfet, Enhancement mode P-channel, with pin numbering Gate[1], Drain[2], Source[3], Drain[4]). This pin numbering will almost certainly be wrong, but we'll fix that as we define the new part. Copy this symbol into your working library, rename it as "TPS1120" and close Transistor.olb. Then double-click the new part bringing it into the parts editor; it should look like the following.

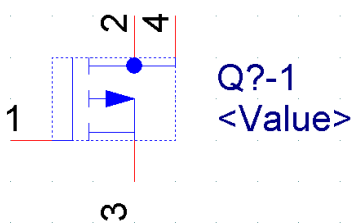


⁵Quoted from OrCAD Capture User Guide, ch. 3, *Opening and developing part libraries*.

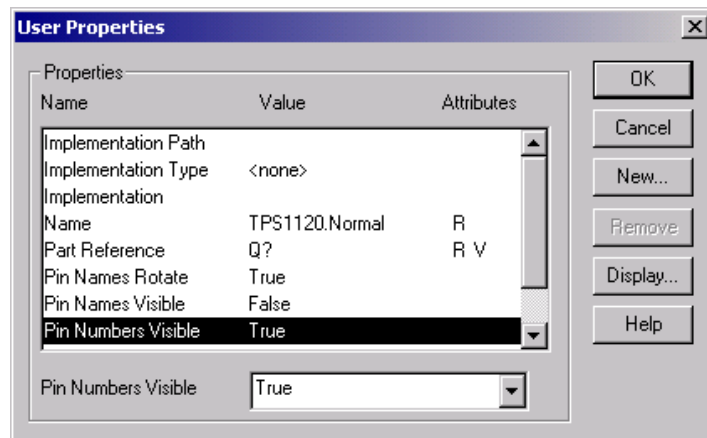
The instance designator “Q?” shows this to be a part having only one logical part per package and pin numbers are missing, so we’ll need to make some edits. Select **Options** → **Package Properties** to bring up the **Edit Parts Properties** dialog. Change the properties to those shown.



Click **OK** to go back to the part creation workspace.

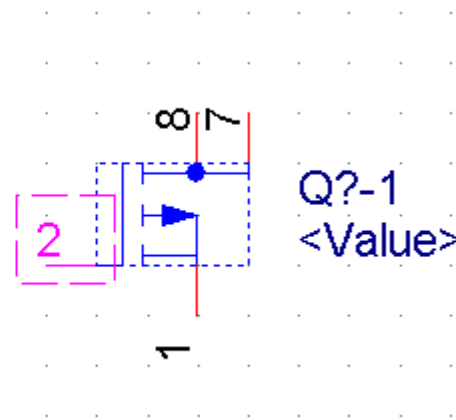
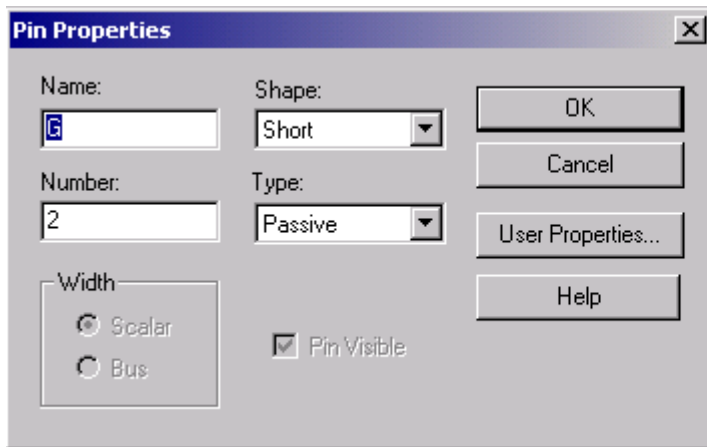


Notice how the part has changed: pin numbers are now visible (and wrong); the instance designator now includes “1” showing that we are editing the first of two instances (had we specified alphabetic, it would be displaying “A” instead of “1”). You can also see how the original entry reflects the pin sequence GDSD for Gate(1), Drain(2), Source(3), Drain(4). Re-number the part so it agrees with the datasheet schematic’s first instance, labeling the pins G for gate, S for source, and D for drain, in case they’re not already so named.

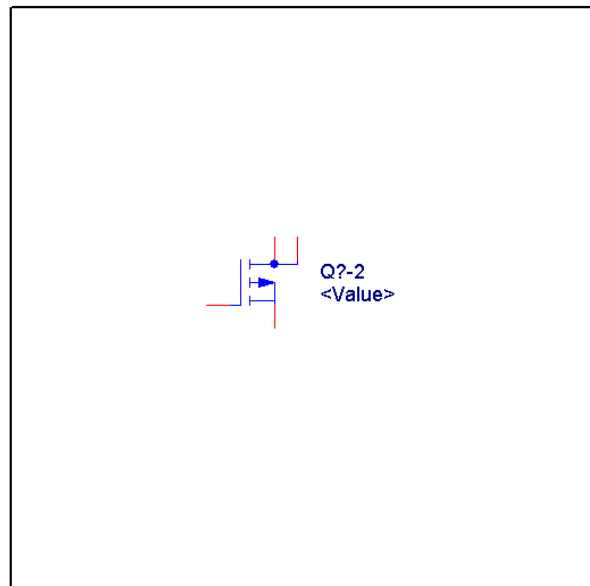
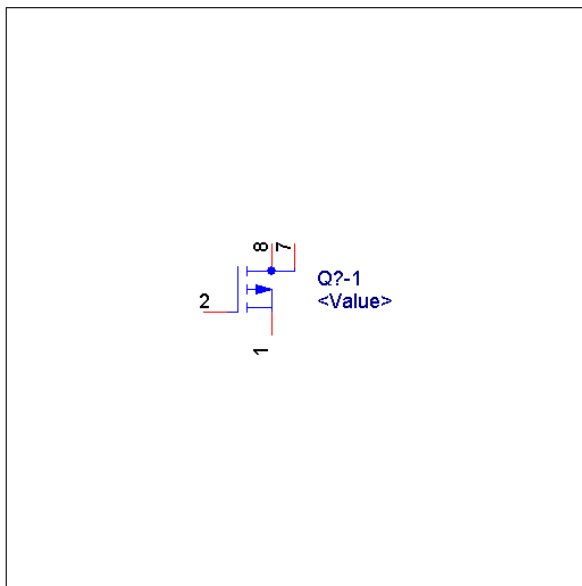


Note: if pin numbers are not visible, make them visible now by setting the *Pin Numbers Visible* property in the **User Property** dialog as shown on the left(**Options** → **Part Properties**).

An example is shown for the gate connection to pin-2 of instance 1 after changing the pin number from 1 to 2 (double-click the pin to bring up the **Pin Properties** dialog for each pin). The resulting fully edited part is shown on the right. Remember, names must be unique for each physical pin per part. As shown, G is assigned to pin-2, S for pin-1, D1 for pin-8 and D2 for pin-7.

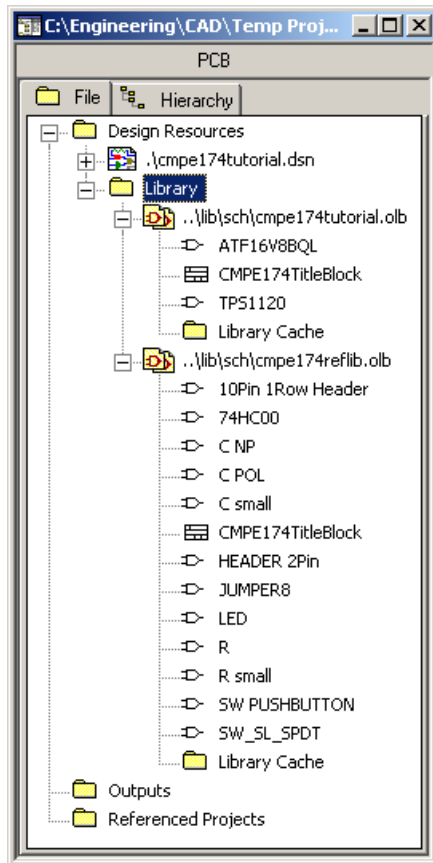


Select which logical package to edit by menu sequence **View** → **Package** to view all logical packages (only two in this case). The second instance shows no pin numbers since they haven't yet been defined (still blank).



Double-click on the second logical part, Q?-2, and Capture will move it to the parts editor so pin numbers can be assigned to it, but using the same names, S, G, D1 and D2. It may be necessary to zoom out to see things, since Capture is not too smart here and often places it off the screen! If necessary, just quickly zoom out with the **O** key until it's visible, position the mouse cursor over it and re-zoom to the center with the **I** key. Proceed to assign additional pin numbers as with the first instance. One word of caution, since this is a homogeneous part, only change the pin numbers and nothing else. After finishing, save the newly edited part.

Why did we specify that pin numbers be visible? The reason is that we might want to choose either of the logical parts based on actual PCB layout constraints, so being able to see them helps in this regard. A good example is the familiar 74HC00, another homogeneous part used in this project having four logical packages. When the schematic is drawn, package choice is largely arbitrary, since we don't yet know any of the related wiring details that will naturally appear during the PCB layout phase later on.



Adding a Reference Library - You will find it much easier to copy parts from the tutorial reference library if you explicitly add it to your project. Do this by first highlighting the virtual library (shown in Blue highlighting) in the Project Manager using the right mouse button. Proceed to select **Add File** from the associated pop-up menu. You should see a dialog box titled, **Add File to Project Folder – Library**. Navigate to your project's ...Lib\ folder and add library EE174REFLIB.OLB and expand both libraries as shown. Copy all the parts (plus the title block) from the reference library over to your library. Note that click and drag will move rather than copy, so it can't be used here. In the example shown, I moved only the titleblock, but rather than do these one at a time, hold the CTL key down, select all you want to copy, and use hot-key CTL+C (or right-click and choose *Copy*) to put them on the clipboard. Follow by highlighting the EE174tutorial target library and paste using hot-key CTL-V (or right-click and choose *Paste*). After successfully doing this, remove the reference library from the project manager (highlight EE174reflib.olb and hit the delete key). Be sure to save these new additions to your library (right-click EE174reflib.olb and choose *Save*). After finishing, collapse the virtual library folder.

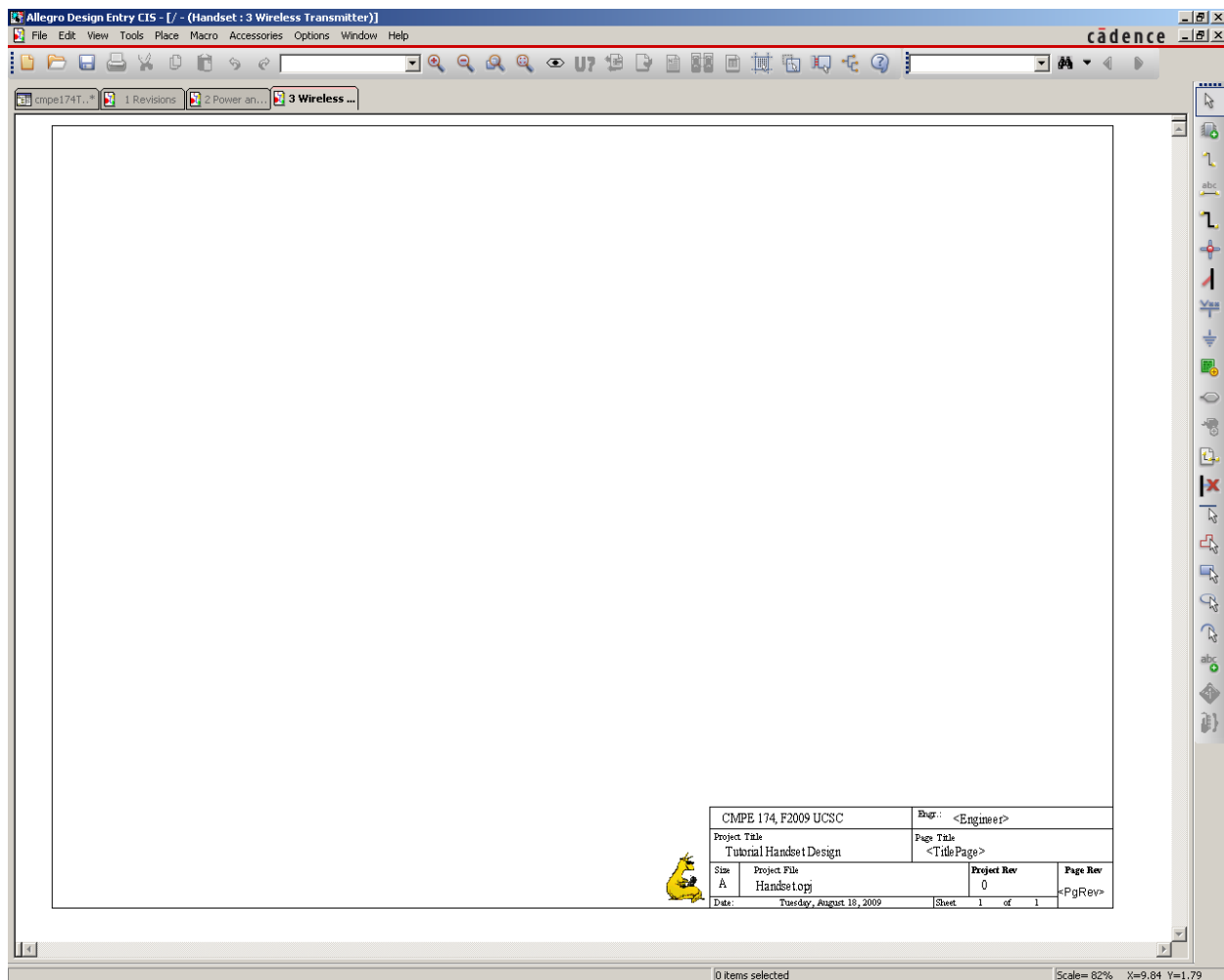
Proceed to create the HT12 and LM317 parts using the techniques learned above. The LM317 is a classic adjustable linear regulator available in several different packages, each with different pinouts; be sure to use the SOT223 package, *not the TO-220*. Print out relevant pages from these two datasheets and place them in your engineering notebook as you go along. Keeping your notebook organized and current is a great help – especially as projects increase in scope and size. Confirm that all necessary parts used in the project are in your EE174tutorial.olb library before proceeding.

CREATING HETEROGENOUS PARTS -What about making schematic symbols for *really* complex parts, like an FPGA? Although we won't treat this topic in this introductory tutorial, Capture allows us to create complex hierarchical and heterogenous parts. Heterogenous parts have dissimilar internal blocks that are not simple replications of each other. I have included an example of a heterogenous part for one of the Altera Cyclone II FPGA's, the EP2C5T144. This part has 144 pins, many of them are power supply connects. Proper use of heterogenous parts can make graphic schematics *prima-facie much* easier to read and, thus, significantly improve their self-documenting features. Hierarchical parts are abstractions of actual internal circuits composed of wired heterogenous or homogeneous parts; this topic is easy to learn, but won't be treated further in this tutorial.

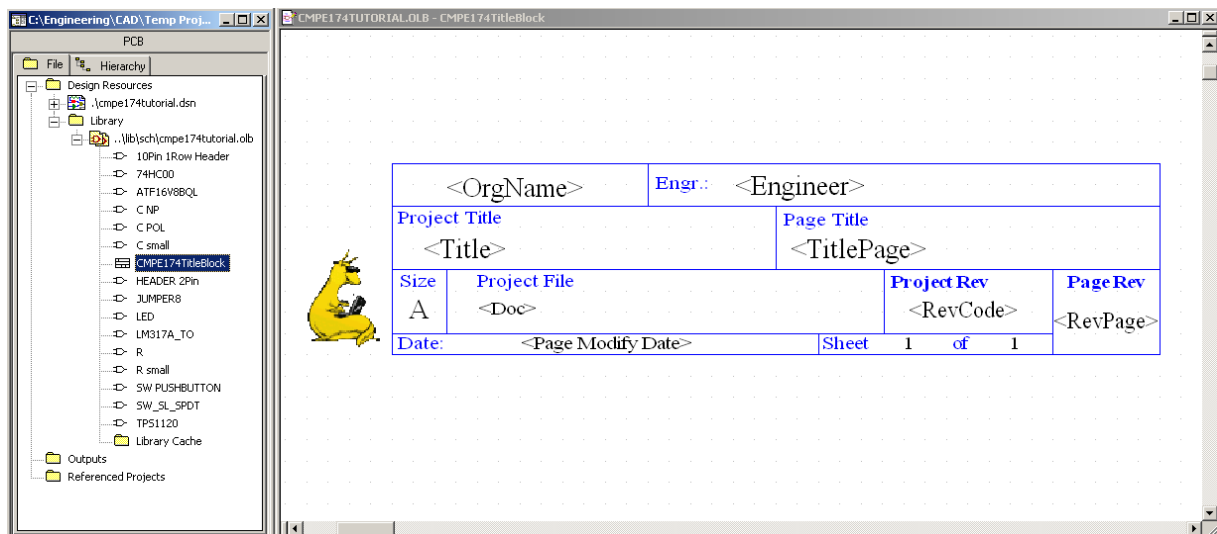
3.4 DRAWING SCHEMATICS

By now you have probably noticed that using the right mouse button brings up context-dependent options. This desirable GUI architecture is prevalent in both Capture and Allegro, and we will use it extensively while drawing schematics.

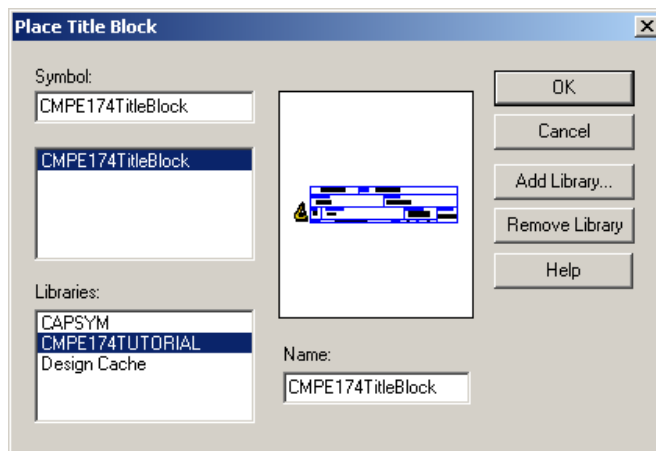
Schematics are created by first placing and positioning necessary part symbols followed by interconnecting or “routing” them to completely “capture” or express the electronic design. I will show you this for the third schematic page, **Wireless Transmitter**. Activate it and maximize it to full-screen. Capture uses **I** and **O** as hotkeys to zoom in and out. Zoom out using the **O** key until the entire schematic page fills the whole screen as shown below.



Working With Title Blocks - Notice that several title block fields are already filled in based on fields you previously entered in the Design Template / Title Block dialog. However, three fields, delimited with brackets, haven't yet been defined. As I mentioned earlier, title blocks can also be added manually. Knowing how to do this is quite useful, particularly if we want to use a different title block or edit features of the existing one and then re-insert it. I will show you how this done, but first, let's see what the title block looks like in the library.

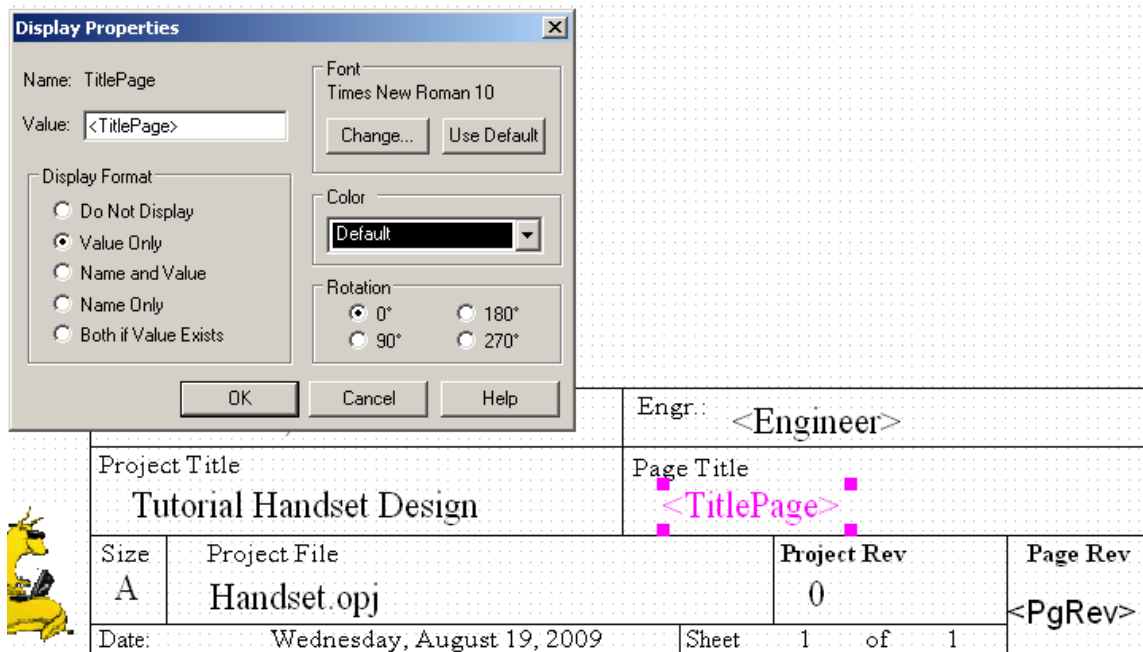


Double-click the *EE174Titleblock* in your tutorial library and Capture will bring it up in an editing window just like the parts you worked with earlier. Notice several things: 1. Editable fields are indicated in brackets. Each of these is a *property* and can be edited the same way part properties are. 2. The title block has a different icon than other parts in the library, and is assigned when it is created (by right-clicking the library and choosing **New Symbol** rather than New Part). For now, just close the edit window, go back to the schematic page and select the entire title block, not an individual property field within it, and delete it (it should be entirely highlighted in purple; follow by pressing the Delete key).




Manually add a new one by following menu sequence **Place** → **Title Block...** This will bring up the *Place Title Block* dialog. You may have to add your EE174Tutorial library. As shown, the libraries currently added include the Design Cache (always there by default) and CAPSYM, one of the Capture symbol libraries that contains many title blocks, among other symbols. Choose the EE174TitleBlock from your tutorial library, press OK and carefully place it in the lower right corner as before.


Since Capture treats title blocks much like parts, it's not surprising they appear in OLB libraries along with schematic and part symbols. After placement, customize the three undefined property fields and re-position as necessary for appearance. Set the Project Rev to "0.1" and center it; replace the <Engineer> field with your name; set the PageRev to "0.0" and center it. These undefined properties can be edited by double-clicking or first selecting with the right mouse button and choosing **Edit Properties...** An example is shown below for the <TitlePage>, where you would then enter the new Value as "Wireless Transmitter". Title blocks can also be edited through the **properties spread-sheet** discussed later in the context of parts.



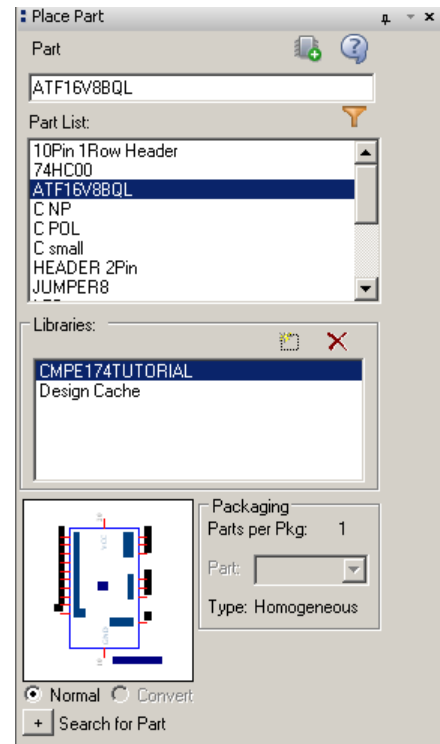
We are now ready to start placing electrical components. Parts are selected and moved onto the active page by

clicking the **Place Part** icon  seen on the vertical toolbar at the far right of the screen. Be sure snap-to-grid is ON to avoid later connection problems.

Note: Capture generally provides three ways to invoke actions like this: clicking on an icon, choosing from a pull-down menu or using a hot-key. For example **Place Part** can be invoked by the icon noted above, the menu selection sequence **Place** → **Part...**, or the hot-key **SHIFT+P**. You should become familiar with these and learn to quickly take advantage of the one that suits you best. The very earliest versions of Orcad Capture only supported hot-keys, naturally resulting in a longer learning curve. Now, some of us avoid learning to use the hotkeys when they are actually faster and more efficient than the mouse!

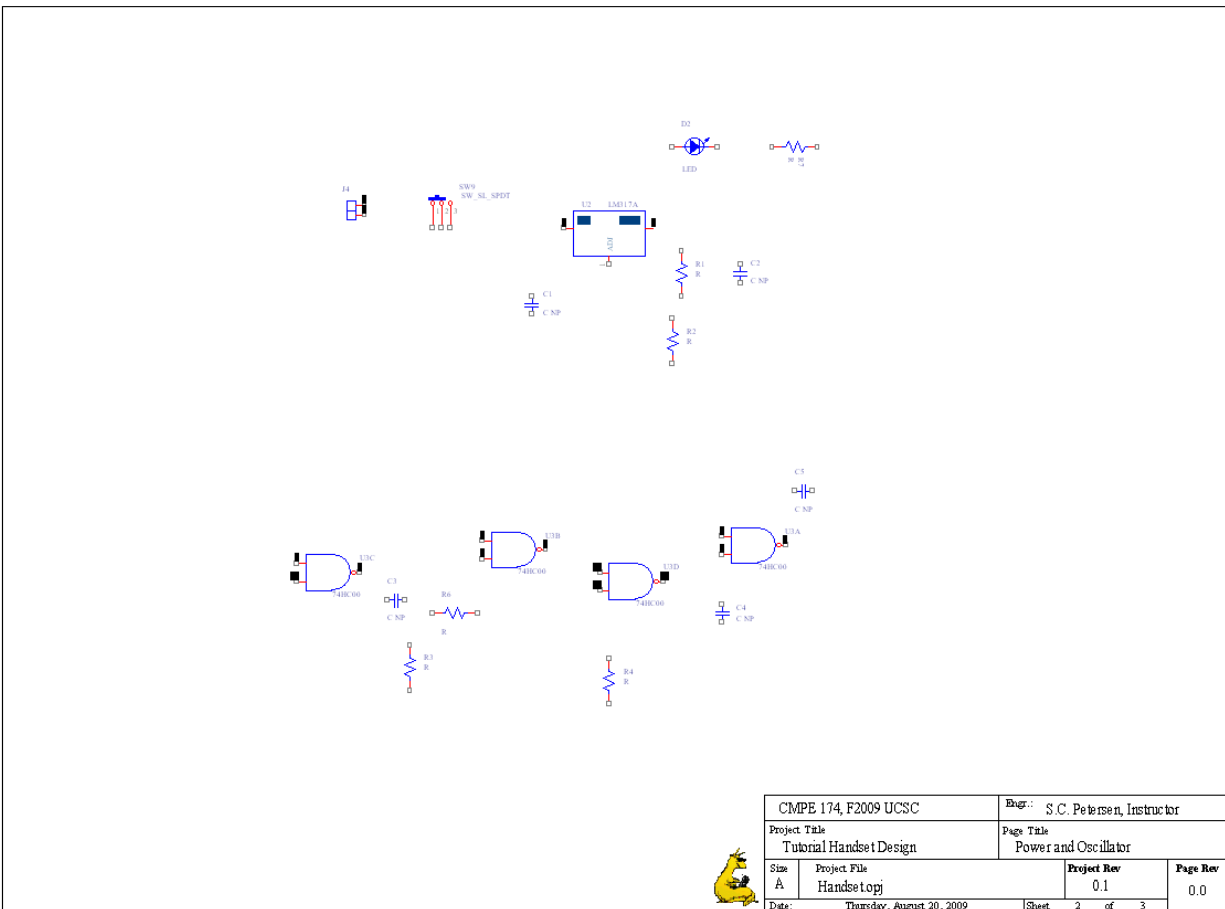
You will then get a dialog for selecting parts to place on your schematic. If the EE174tutorial library is not present, press the **Add Library...** button  and navigate to it to add into the local list of working libraries; the red “X” icon similarly deletes a listed library.

Shown is the new tutorial library along with the **Design Cache**, which we will discuss shortly. Choose the ATF16V8BQL and place it on your multi-page schematic using the completed hardcopy handout of the Wireless Baseband Encoder as a guide (place by double-clicking the part or pressing the Place Part icon at the top of the dialog; be sure to right-click and choose End to finish each placement or select another part immediately). Continue to add all the parts seen on pages 2 (Power and Oscillator) and 3 (Wireless Transmitter) without paying too much attention to exact placement. Close the Place Port dialog by clicking on the “x” in the upper-right corner. Note that since this dialog is dockable, you can “unhook” it from the right-side and move it where you like.

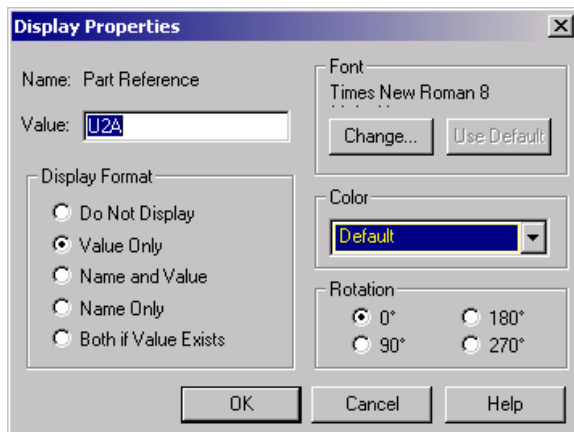


In addition to **I** and **O** keys for zooming in and out, you can also use the **R** key to rotate a symbol and the **C** key to center the screen at the current mouse position. Hence, it is possible to effectively pan around using the mouse and C keys together.

At this step, your second schematic page should look something like the following:

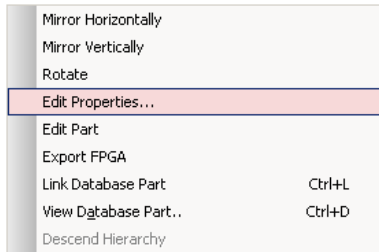


Note: You should see symbols with input pins denoted with arrows to more clearly denote their function. Along with each part's graphic symbol is a set of visible and invisible embedded properties. Capture provides several ways to edit these: individually, by double-clicking on those that are visible, like instance designators and part values; collectively, in a powerful spreadsheet format that makes keeping track of them all relatively easy. To show you how this works, edit the four instance designators to agree with the handout schematic by simply double-clicking on each one to bring up the **Display Properties** dialog. For example, double-clicking on part reference (or instance) U2A brings up the following dialog (yours might just already be correct, since reference designators are assigned sequentially based on when parts happen to be place – click away to learn how this is done):

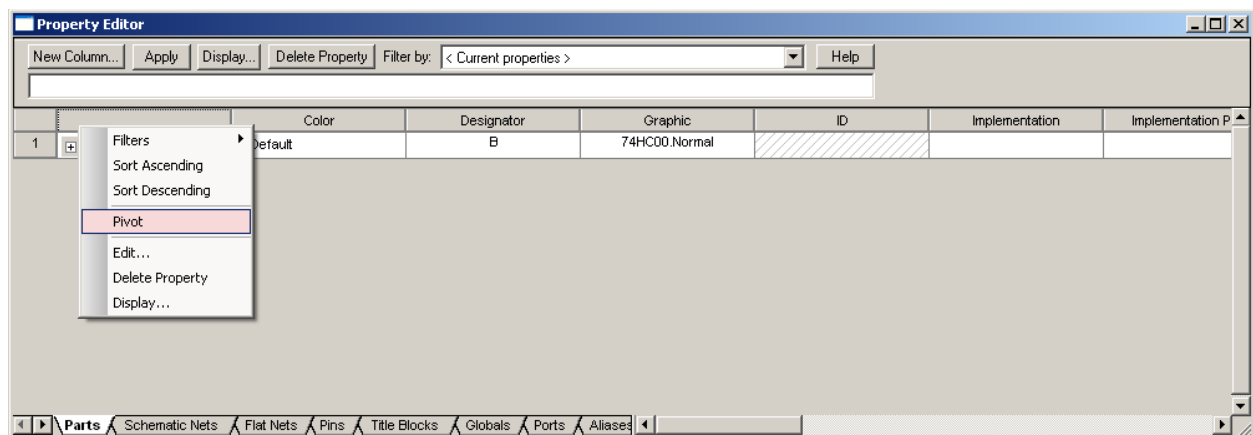


U2A can then be edited directly and changed to U3A in the text box shown (not yet edited). Note the **Display Format** options, where you can also set whether a property should be displayed or not. For example, we have four Nand gates all within the same single 74HC00 package. Rather than display all four, we can display only one. I like to use the convention of displaying this for the first instance only, in this case for U3A (*c.f.* the handout schematic). If the parts are widely separated, making the Part Reference visible aspect.

Editing more than one property at a time, working with otherwise invisible properties, or working with several parts' properties at once, the spreadsheet interface must be used. Select a single part using the left mouse button followed by the right mouse button. For multiple parts, hold the CNTRL key down while selecting each single part in the group; they will each highlight progressively. Follow with the usual right mouse click to show the context-sensitive pop-up menu as follows (only part of this menu is shown; its quite long).



Choose **Edit Properties...** This will bring up the spreadsheet properties editor.



The screen shot above shows the spreadsheet as it will probably look when first invoked with properties arranged in horizontal fashion across the top of the screen (row display), one line for each part (here only a single part was selected). You should now *pivot* this and show it as a single column (column display) by right-clicking the mouse anywhere on the row containing the gray rectangular area just above the top edge of the pop-up menu. Choosing **Pivot** will toggle between the two formats. Changing from row to column format is MUCH easier to view and edit, especially for single parts. Always choose the best presentation format, column or row depending on what you want to do.

TIP: Note the various tabs along the bottom. These act as filters allowing us to see only wanted subsets of the otherwise large group of available properties determined by a master filter setting, *Filter by*, at the very top. The tabs used most often are *Parts* and *Pins*. If the spread-sheet appears with the wrong filtered subset, select them manually from these tabs and check the correct master filter setting. Available properties are also affected by the context. Selecting a part symbol and then choosing the Title Blocks is a meaningless action, since a part is not a title block. Doing this (try it) will result in a spreadsheet with “No Object Selected”.

Property Editor

New Row... Apply Display... Delete Property Filter by: < Current properties >


U3A

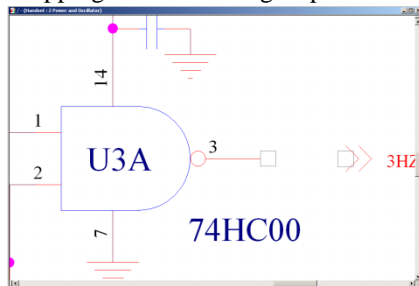
A	
	Handset : 2 Power an
Color	Default
Designator	A
Graphic	74HC00.Normal
ID	
Implementation	
Implementation Path	
Implementation Type	<none>
Location X-Coordinate	925
Location Y-Coordinate	560
Name	INS14174723
Part Reference	U3A
PCB Footprint	
Power Pins Visible	<input checked="" type="checkbox"/>
Primitive	DEFAULT
Reference	U3
Source Library	C:\ACADEMIC\UCSC\20...
Source Package	74HC00
Source Part	74HC00.Normal
Value	74HC00

Parts Schematic Nets Flat Nets Pins Title Blocks Globals Ports Aliases

The resulting pivoted column display view is shown above, making it easy to now edit **Part Reference** U2A by changing the value shown to U3A (use the mouse to shift the focus to the text field as I have done). Know how to toggle between column and row modes, since you will frequently use this feature. I have also checked the *Power Pins Visible* property, since we will want to visually show the power pins on this part instance (the other three in this homogenous part will remain invisible, since they are merely redundant). When finished, close this local window by clicking on its "X" icon in the upper right corner (not shown, since I clipped this screen shot).

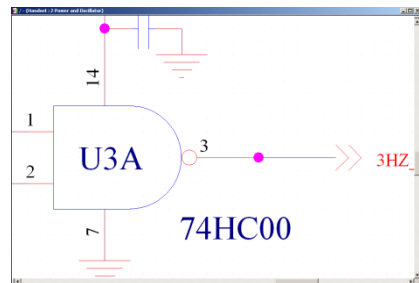
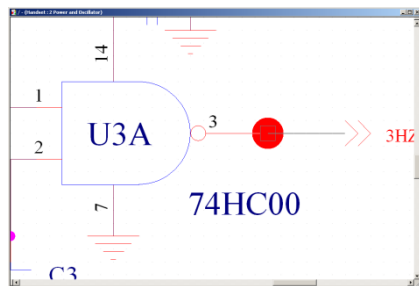
At this point, we could also do what we did earlier by accessing the **Display Properties** dialog directly for the Value property, but from within the spreadsheet by selecting the property name with the left mouse button, then the right mouse button and choosing Display from the popup menu (the property must, of course, be editable). For example to make the last property in the list, Value = 74HC00, invisible, right-click on **Value** → **Display...** to see the Display Properties dialog box. Try this on several different properties to get the hang of it.

MAKING CONNECTIONS – Use **Place Wire**  to draw connections between electrical components. The only way to learn wiring is to actually do it. Capture uses a vertex recognition scheme to join connectivity between wires or wires and connection points on components. Thus, wires that visually appear to be connected may not actually be so, *unless their vertex points are aligned*. This is a common problem for beginners, and the best way to avoid it is to LEAVE SNAP-TO-GRID ON. That way, wires will always begin and end on grid points; parts will always be placed so their wire points align with grid points; and all will be well (we hope). Wires always begin and end by snapping to the nearest grid points.

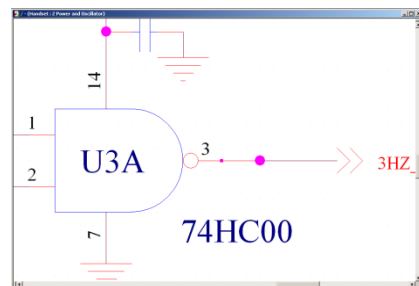


For example, suppose we want to connect U3A, pin3 to the double-arrow off-sheet connector as shown. The two gray boxes indicate where the vertex points are for each connection. I chose to begin by first clicking on the right-most box, causing the wire to snap to the closeset grid point – namely the one enclosed by the box, and then proceeded to drag the new wire over to the left gray box for pin3.

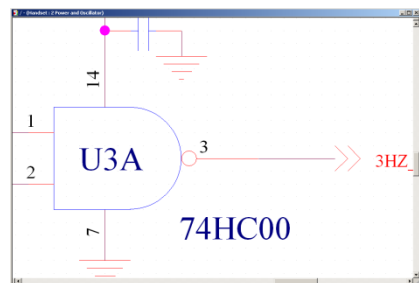
When I got within snapping distance of pin3's vertex point, Capture jumped the wire to this point and displayed a solid **red dot**. This is really the *only* definitive way to tell identify a real connection. Therefore, always look for the tell-tale red dot to appear whenever two wiring vertexes align. **Note:** this feature is enabled only when the *Wire Drag* checkbox is checked on the Preferences / Miscellaneous dialog (see page 3 above where this was done earlier).



What happens if we overshoot and drag the wire too far? If you try this, you'll see the red dot still appears, since the first wire ended normally on the end point, but a new segment immediately began and now extends some distance beyond it. After concluding the Place Wire mode (right-click and End, or press Esc.) a purple connection dot oddly appears indicating an electrical junction of three wires: the pin and the two segments. The excessive overlapping segment isn't visible since it's overlaid on pin3! Fix this by selecting and deleting the dangling wire segment.

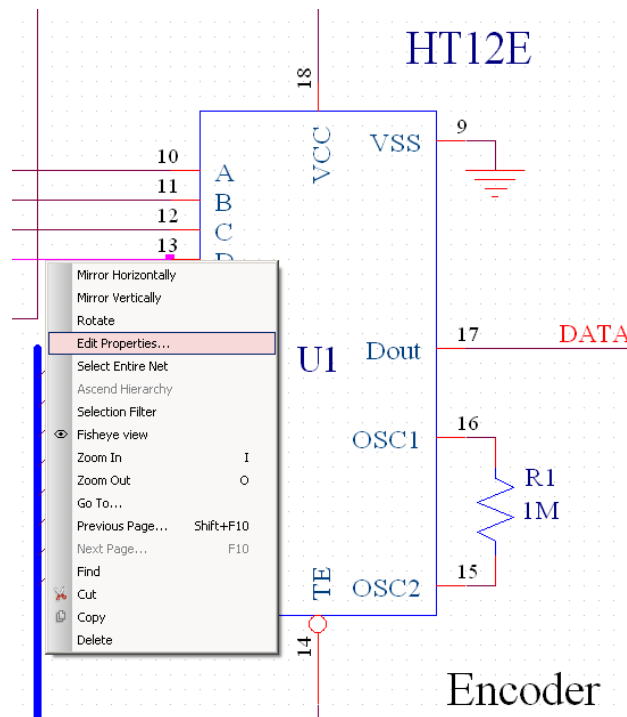


To convince you, I've selected the overlapping segment just left of the connection point where it can be clearly seen because its now highlighted. Simply delete it and the purple junction dot will disappear. Note the tiny dot merely demarks the line's end, not a junction.

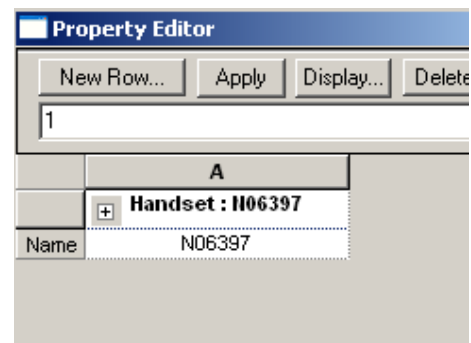


This is the way it should look. I've taken the time to show you how this works and explain it, since beginners tend to commit this minor error frequently.


NETS, CONNECTIONS AND NET NAMES – Capture assigns every connection a unique **net name** at the time of connection, for example N20126 is a unique name for “net number 20126”. The name of any net can be viewed by first selecting it with the mouse (initially hold the CTL key down at the same time if you don’t want it to move around) followed by the right mouse key to bring up a pop-up menu and then selecting **Properties**. Capture will show you the familiar spreadsheet with the connection’s single property: its net name. Don’t be tempted to change the net name to something else; it can’t be done here! Capture will allow you to edit the text field and close the spreadsheet, but the net’s name will stubbornly remain unchanged (the next section describes how to do this).

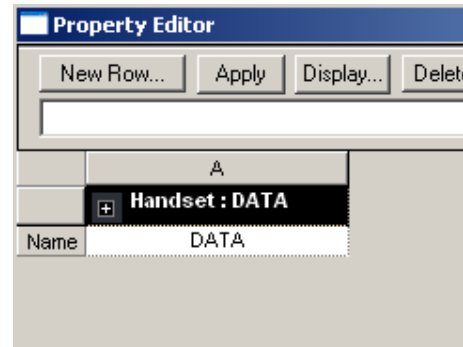
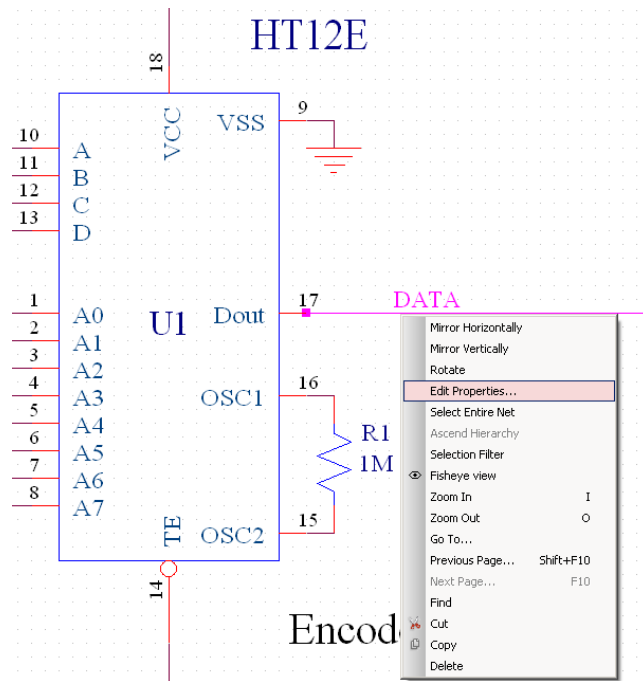


For example, the left figure shows the wire connecting to pin-13, U1 (the HT12E) selected for editing. Choosing **Edit Properties...** will then bring up the spreadsheet showing the net’s Capture assigned name to be N06397.



The idea of a “net” arises from the historical network of connections comprising an electronic design. Hence, we call wires or connections “nets”. Some rules you should be aware of: nets having different names cannot be wired together or Capture will flag them as shorted. Floating nets are connections that don’t make a complete circuit from any single output to at least one input. Hence, a net tied to an output but never connected to any input(s) would be floating. So would a net connected (or tied) to many inputs but having no output connection (net has no driving source). Outputs can, of course, be tied to many inputs, but outputs cannot be tied together without causing an error, unless they are open-drain or open-collector types. These and other errors will only show up after running the **DRC** (design rule check) at the end to carefully look over your design for errors and produce various warnings for things that are legal electrically, but you may not otherwise be aware of (more on this when we get to the end of this lab!).

CHANGING NET NAMES – Use **Place Net Alias**  to change the name of any single net or to name a bus. An example is the net connected to pin-17, U1 above. After selecting the icon, we get a dialog box asking for the new net's name; enter it, click **OK** and then select the net to rename with the mouse (try it). You can also highlight the net first, and then select the icon. The new name can be positioned anywhere along the wire you like. Repeating the exercise above, selecting the net connected to U1 pin-17 and observing its properties, we note the name has indeed been changed to "Data".



Use this feature to name important nets; an example is "DATA" above. Aliases aren't needed unless it makes your schematics more understandable or useful during the PCB layout phase.

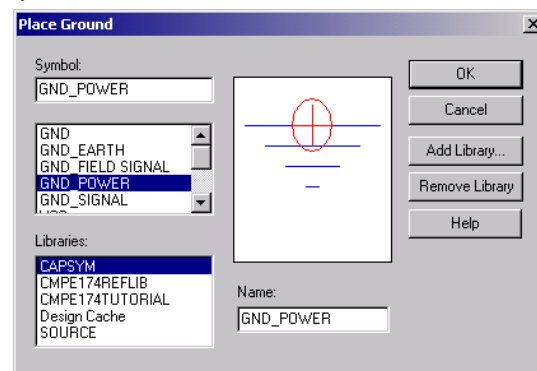
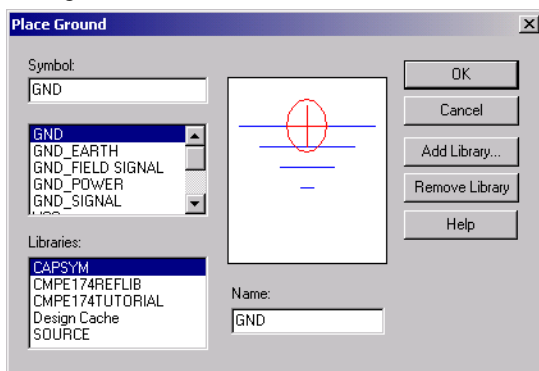
Another menu item particularly useful with nets is **Select Entire Net**. Choosing this will highlight every connection having the same net name.

POWER AND GROUND CONNECTIONS – All circuit designs having active components require power from a suitable power supply. Typically, the supply is a voltage source of some kind designed, ideally, to output a stable and noise-free supply of current to make everything go. How we show the necessary nets for distributing power is largely a self-documentation issue: Drawing nets from any power supply block to every circuit using it usually leads to an unnecessarily confusing mess; this is compounded when more than one supply is present in a system. Consequently, it has become standard practice to employ special symbols to specifically indicate distributed power networks without having to actually show the interconnections. Think about this for a moment. Power distribution actually involves two nets, one for the source of current and one for the returning current. Capture has a group of built-in symbols for connecting nets known as *power* and *ground* found in the CAPSYM.OLB library.

Capture has two icons on the right-most toolbar for accessing these power and ground symbols. Both icons bring up the same dialog box. The only difference is the label will say “Place Ground” instead of “Place Power”. I will discuss both of these in turn.



GND Connections - Several things should be borne in mind, especially when using the ground symbols. *First*, for historical reasons, the common node is usually called “ground” or “earth” and is simply the name given to a particular electrical node designated to be the reference or “zero volt” node, from which all other voltages are referred. Remember that voltage is the difference of potential between two nodes; it cannot exist at a single node. *Second*, we must be very careful here, since Capture provides several identical looking ground symbols that are defined by different net names, namely “Ground” and “Ground_Power”; “GND_FIELD SIGNAL” and “GND_SIGNAL”. Unless you know exactly what you’re doing DO NOT mix these symbols even though they look alike, since Capture identifies a net by its name – NOT its graphic symbol. Hence, you may innocently select the same ground symbol at different times, but inadvertently confuse the net names. To see what I’m talking about, select the GND tool and note that GND and GND_POWER are represented by the same symbol, as shown below. In this lab, just use the GND symbol exclusively. Finally, connecting any of these power symbols to existing nets will change those net’s names to that associated with the symbol.



	"0"
	"GND"
	"GND_EARTH"
	"GND_FIELD SIGNAL"
	"GND_POWER"
	"GND_SIGNAL"

All possible default ground symbols and their assigned net names are shown to the right. In mixed-signal designs, those having digital and analog circuits, we frequently use two different symbols, one for the analog ground and one for the digital ground. Since they are both typically powered from one or more power supplies having the *same* zero-volt reference node, the two symbols *must* eventually connect together somewhere. For example, if we use the “GND_SIGNAL” net for the analog common node and “GND” for the digital common node, somewhere on the schematic, usually near the power supply, they would be connected together as shown below.



You might be wondering why this is done, since after all they’re really same anyway. The purpose improves the self-documentation by drawing attention to the fact that we, as circuit designers, want to

control how the return currents physically distribute on the finished

PCB to deliberately minimize unwanted parasitic interactions between the digital and analog sub-systems otherwise sharing the same common ground returns. Further discussion about this important topic is well beyond the scope of

this lab, but you as an engineer should know it exists, and be prepared to investigate it more deeply should the need arise in practice.



Power Symbols – We can do the same thing with the source network as the return network, and Capture provides us with a set of symbols specially meant for this purpose:



How these several “VCC” symbols are used is entirely up to you. I like to employ the round symbol, either VCC or VCC_CIRCLE to denote the master output point from any particular power supply (since the names are really net aliases these two symbols are otherwise visually equivalent), and the arrow symbol to “point to” the master distribution point. Unlike the ground symbols, these node names are meant to be re-defined at the time of placement. In the tutorial project, I assigned the circle and arrow symbols to the “+4.5V” net. This is the regulated voltage supplied by the LM317 linear regulator on page-1.

JUNCTIONS, NOCONNECTS AND TEXT



Capture allows wires to cross without regarding their intersection as a connection, or junction, unless you deliberately start or end a connection on a wire. These points, where 3 or more wires join, are always flagged graphically with magenta dots. If two wires cross without a junction, they don’t connect, but they can be made to connect by using the **Place Junction** tool. I have noticed that occasionally Capture will fail to place a junction where it should otherwise be; placing it manually is then called for.



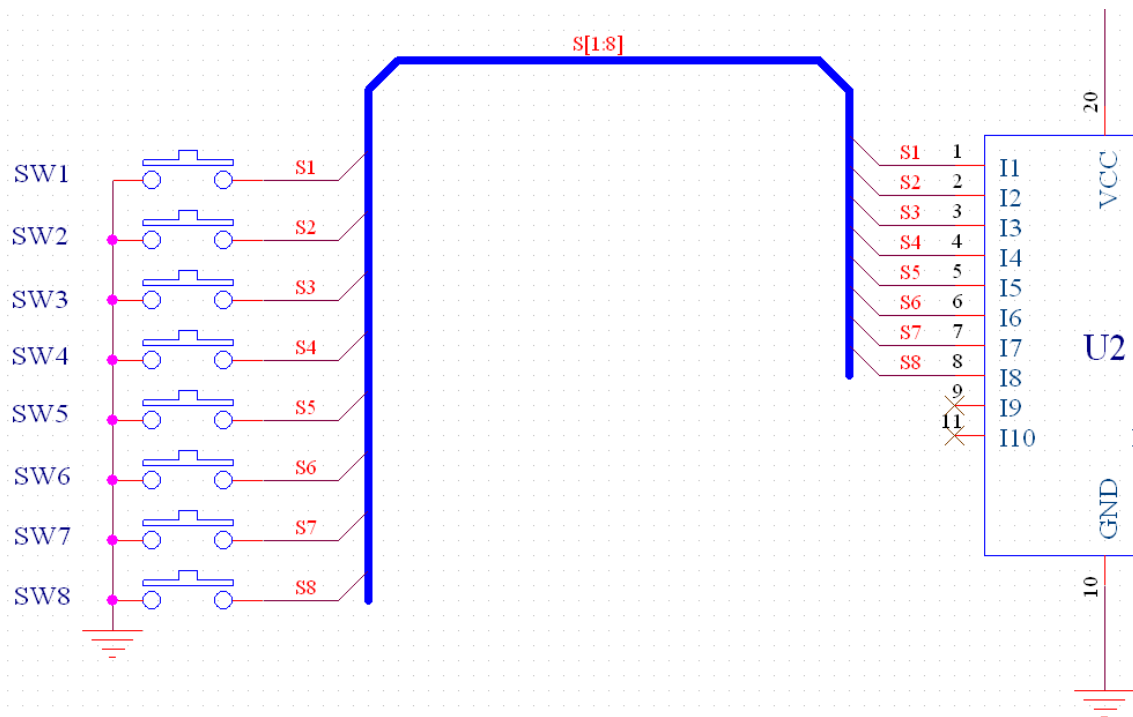
Use *no-Connects* to explicitly flag pins deliberately left unconnected. This serves as a visual reminder they weren’t forgotten, and to flag the DRC (design rule checker) not to generate an unconnected net warning. Assign them using the **Place No Connect** tool. These “X” symbols will disappear if you later connect a net to them, but will subsequently cause a netlist compilation error and must be removed. The only way to do this is by unchecking the pin’s *Is No Connect* property using the spreadsheet property editor for the particular part involved. Access the spreadsheet in the usual way but be sure to select the **Pins** tab at the bottom. Several of these exist on the ATF16V8BQL part.



Add text by using the **Place Text** tool. Selecting it will bring up a dialog box that allows you to specify font, size and appearance. For precise placement, turn off the snap-to-grid, but remember to turn it back on afterwards.

WORKING WITH BUSES – Buses are an important topic in engineering design and frequently appear on schematics, so you should know how to work with them. Except for a cursory overview, we'll only concentrate on depicting them in Capture.

A bus is any group of wires meant to carry closely related signals. For example, in cpu-based digital systems, where they are probably most prevalent, address, data and control signals are carried over buses. Capture treats buses much like individual wires: they can be drawn, named and connected to another bus having the same name. Once a bus is created, connections are made to it by nets having assigned aliases that must be within a specified range determined by how the bus is named. In this tutorial project, I have used buses in two cases, more for illustration purposes than because we really need them. The first interconnects the set of eight command buttons to the ATF16V8BQL PAL chip. On the original schematic I wired SW1 through SW8 directly to the PAL (U2), but here I've wired them through a simple bus. Similarly, J1, the set of eight 2-pin address jumpers is connected to the HT12 Encoder chip via another bus. I've included a screen shot for the switch bus below.



Use the **Place Bus** tool to add buses as thick blue lines. I like to draw them with 45 degree corners by holding down the Shift key when first starting a segment, but they can equally be drawn with right angles. As shown above, the thick blue line represents the bus carrying eight related signals, named S1, S2, ..., S8. Unlike wires, buses must be specifically named according to a formal alias bus naming notation that clearly abbreviates the group of related signals defining the collection of nets carried. In this example, the switch bus is named "S[1:8]", standing for the group of eight nets carrying signals S1 through S8. Each signal connected to a bus, therefore, must bear an alias falling within this range. Hence, the switch bus is named "S" (may be any combination of letters and numbers, but cannot end in a number) and carries an enumerated set of wires beginning with 1, for "S1" and ending with 8, for "S8". Any wires connected to this particular bus must be assigned at least one of these eight possible names.



Wires cannot be directly connected to a bus, but must be interfaced using short 45 degree wire segments known as **Bus Entries**. They are easily placed by selecting the **Place Bus Entry** tool. Once selected, they can be repeatedly placed; rotate them by using the "r" hotkey. Bus entries on opposite sides of a bus can be named differently and not short at the apparent common vertex on the bus itself.

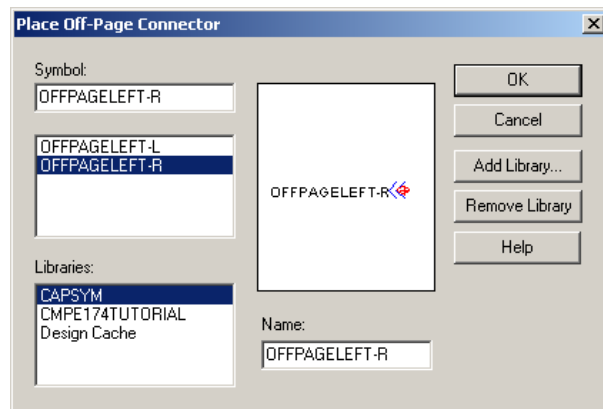
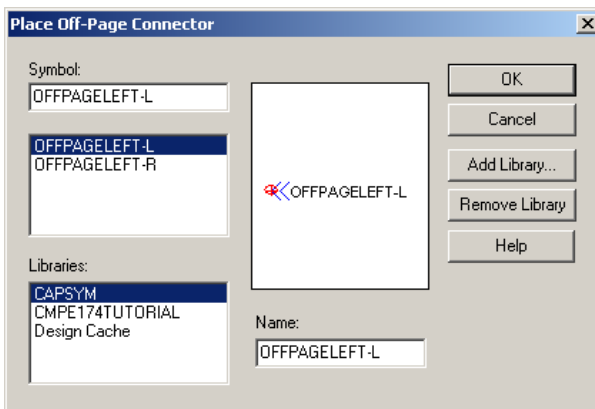
Working with Multipage Schematics – Dividing electronic design projects into multiple schematic pages can prima-facie enhance perception of the underlying engineering design. In this tutorial I have somewhat artificially divided the design into two pages, since the entire schematic originally fit nicely on one page. My purpose, of course, is pedantic. I want to show how schematics drawn over several pages should be drawn and how to properly notate interconnections between them. Even here, though, proper segmentation is important to good self-documentation. Placing significant design groups, like I/O or power, on separate sheets lends modular elegance and understanding to your overall engineering designs. When faced with a project too complex to fit on one page, think carefully about how to spread it over two or more additional pages before arbitrarily moving things about.

Interconnectivity between schematic sheets within a group of schematics is really quite easy. Basically, *Capture* considers any nets having the same name on multiple pages to be wired together overall. Clearly, then, we must deliberately name nets and buses to realize wanted inter-sheet connectivity, since as we draw on separate pages, Capture automatically assigns different default net names. Buses are different; they must be explicitly named in any case, so naming them identically on different pages easily defines them to represent the same bus.

Capture provides us with a powerful set of symbols meant primarily for use with inter-sheet connections, but they can also be used intra-sheet. These symbols are all found in the CAPSYM.OLB library. Several tools conveniently filter subsets from this library.

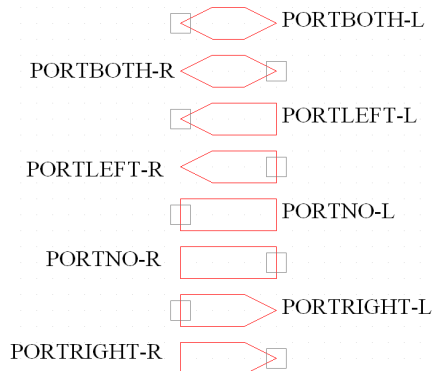


The **Place off-page connector** tool accesses two useful symbols: *OFFPAGELEFT-L* and *OFFPAGERIGHT-R*. These can be used with wires or buses. Their primary intended use is to indicate interconnections between pages, but they can be used on the same page if it improves self-documentation. There is absolutely no distinction between them, other than which side of the double-arrow the vertex appears on. The two names are actually aliases, meant to be renamed when the object is placed. I like the convention that uses arrow directions not to indicate “going off page” or “coming on page”, but to indicate *signal direction*. For example, on page 2, net “CLK_ENABLE” indicates an off-page output signal from U2, while net “3HZ_CLK” indicates an on-page input signal to U1. Formally, though, neither of these are absolutely necessary to express the on or off-page intentions. Merely leaving the net dangling, but naming it, is otherwise sufficient to establish inter-page connectivity; this is poor practice since it diminishes lucid self-documentation. I have seen many designs first created in hardware description languages (HDL’s), and subsequently expressed graphically as hierarchical block diagrams, often suffering from this deficiency, since the engineer creating this largely useless “schematic” knows little about self-documentation. The result, from ignorance or expediency, leaves them as big rectangular objects surrounded with many dangling named nets. Avoid this poor practice where possible. Finally, whatever nets these two symbols connect to will inherit their assigned alias names.





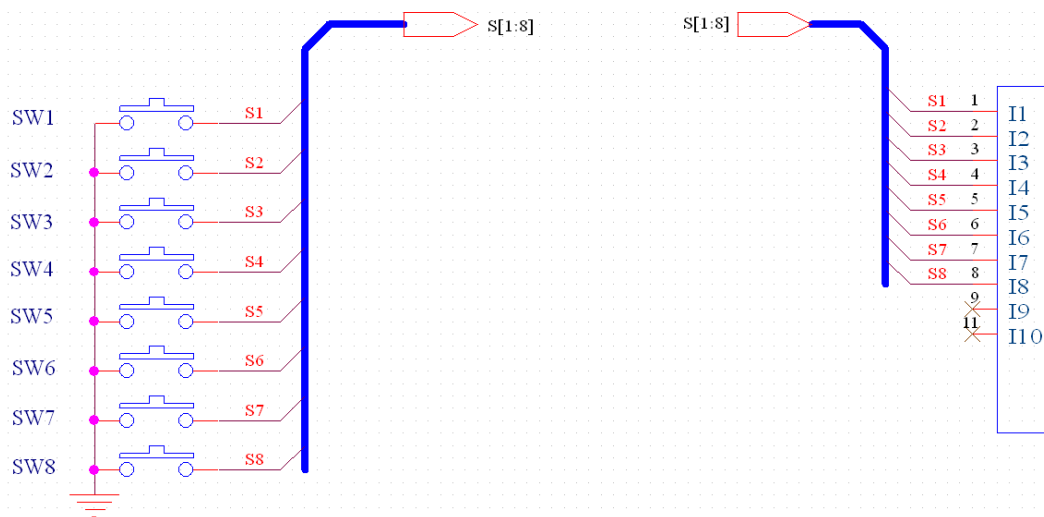
Inter and intra-page connections can also be expressed with *ports* by using the **Place port** tool. This accesses eight port symbols found in the CAPSYM.OLB library that can be connected directly to any bus or wire. These are summarized as follows.



Stylistically, I like to restrict use of these to buses, but the choice is yours. There is one definite difference between ports and off-page connectors. Ports can be also be used with hierarchical parts having direct bus connections, while offpage symbols cannot. I won't pursue this topic further here, except to note it in passing. The following elaboration will link buses specifically with ports, but they could apply equally to wires as well.

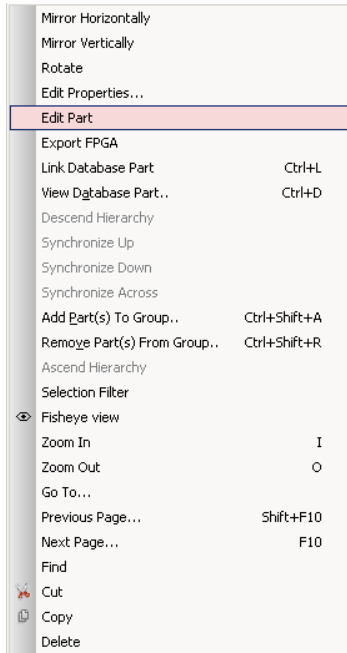
Buses meant to pass signals in either direction are called “bi-directional”, while buses meant for signals flowing in one direction only are “uni-directional”. The first two port symbols above are intended for bi-directional buses, like the address bus in cpu-based digital systems. Observe they are really identical, since rotating either one moves the vertex to the opposite side. This is also true for the uni-directional port symbols, those having a single arrow on one side only. The last set of rectangular port symbols (also identical due to rotation) would be assigned to buses having individual signals flowing in different directions. When renaming these signifiers, they must adhere to the name of the bus they connect.

For example, had we shown the eight push-buttons on another page, two uni-directional port symbols might have been chosen, as shown below. Ostensibly, the switches would be on one page and the chip with its' inputs on another. Note I used the signal flow direction convention. Try this on page two as an exercise and you will discover some unexpected behaviors. When connecting wires, you saw the red dot appear whenever vertexes align. The same thing should happen with ports – but it doesn't *until the port is properly named*. In practice, this means you should first bring the port symbol onto the schematic, name it properly, then move it to the proper connection point.



4. THE DESIGN CACHE

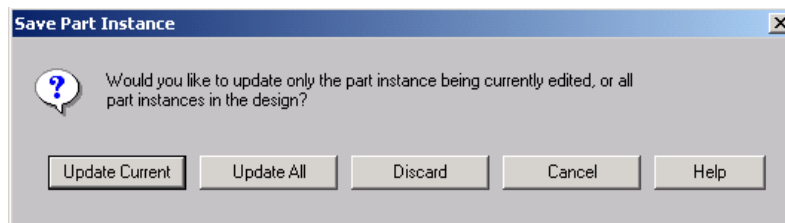
When parts are selected from any source library and placed on your schematic, Capture first copies them to a virtual folder called the **Design Cache**. This provides a way to edit parts placed on schematics for customizing purposes but without affecting the original part in the source library. In other words, Capture always works with a copy of the original part in the cache, but never the original itself. Consider how this works:



Whenever **Edit Part** is selected after highlighting the schematic symbol with the left mouse button, followed by the right mouse button to show the pop-up menu, capture launches the usual library part editor but works only with the part in the cache. This means any changes are updated or reflected within the cache only and NOT the source library the part originally came from. Changes to cached parts will be evident by presence of the telltale “_n” suffix added to their names. For example, if you changed the pin locations on one of the 10-pin headers, whose original part is named, **10Pin 1Row Header**, Capture will flag this by appending “_0” to the first modified variation of it, or **10Pin 1Row Header_0**; subsequent variations would be denoted _1, _2 etc., each reflecting the fact they are indeed different than the original template version. I often use this feature to move pins around one specific chips to make them work better on the schematic (*i.e.* improved self-documentation) *without* affecting other instances of the same part in the same design.

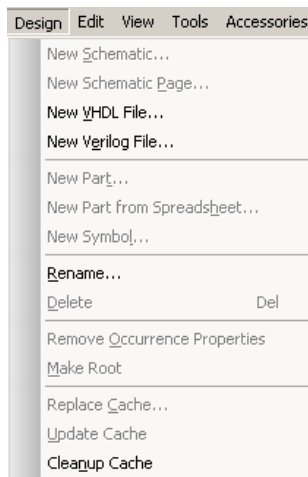
The cache is actually a library exposed by the schematic itself, but unlike a conventional library, we cannot edit parts directly in the cache as you would in, say, your own custom library. Double-clicking a cached part will do nothing! The only way to do this is by selecting them on the schematic itself. In this sense, the schematic *is* the cache!

After editing any cached part, Capture offers us a different set of save options than when saving parts in a normal library. These are summarized in the **Save Part Instance** dialog:



Choose **Update All** to reflect the new changes to any instance of the part anywhere in your design, but choose **Update Current** to restrict changes to only one part – namely the one you’re editing.

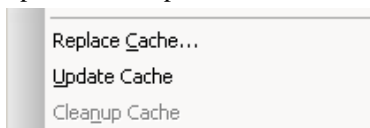
THREE BASIC CACHE OPERATIONS – Changes can be made directly to the cache in three basic ways, all of them accessed by the three choices at the bottom of the **Design** menu list. These are context-sensitive choices.:



Cleanup Cache - After adding, editing and probably deleting parts, the cache will most likely contain parts that are no longer used. “Cleanup” the cache and eliminate these now useless parts by highlighting the cache virtual folder, **Design Cache**, and then selecting the menu sequence **Design** → **Cleanup Cache**. Capture will respond with a question about saving everything first, then proceed to remove obsolete entries. This menu option is also available by highlighting the Design Cache field and right-clicking the mouse.

Contents of the cache display are updated each time it is opened in the Project Manager. Therefore, closing it and reopening will refresh the display (click on the -/+ icon immediately to the left of the Design Cache field).

Every cached part retains a link to its’ source library to enable several possible synchronization operations. This is always done on a per-part basis, so after first highlighting any particular part in the cache, the following two operations are possible.



Using either of these with the option to preserve schematic part properties retains all instance and occurrence properties; this means you won’t lose changes made to pin properties, including those made through the annotation process. Note too that either of these options are also available by the usual

selection highlight and right-click context menu technique. Capture’s help entry regarding is worth reading; an edited portion is reproduced below for convenient reference.⁶

Replace Cache... - This command to replaces the selected part in the design cache, based on its current definition in any library. It can also be used to replace a selected part in the cache with a different part.

Update Cache - This command updates selected parts in the design cache, based on their current definitions in their original libraries, and works whether one or multiple parts are selected. Whenever a part is changed in the design cache, all the parts in the active design will also be changed that share the same part value and library. Part properties are retained, but pin properties are not. If the modified part appears on an open schematic page, the parts on that page do not change until the page is closed and reopened. If you copy pages from one design or library to another, parts displayed on the copied pages may appear different due to differences in each design or library cache. If a part is not already in the destination design cache, Capture will copy it from the source design's cache. Otherwise, it will use the part already present in the destination design's cache.”

If you need to know a part's library of origin, select the part in the project manager, then select Replace Cache from the Design menu. The part name and the library and path will then be listed in the dialog box that appears. Click Cancel to return to the project manager.

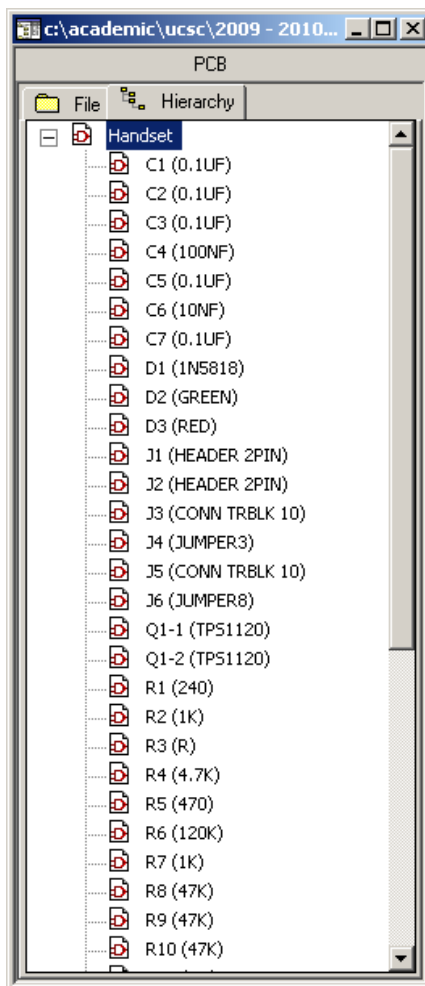
Note: The Replace Cache and Update Cache commands are quite similar. However, there are a couple of significant differences between them. You can modify a part's link to the library (part name, path, and library) with Replace Cache, but not with Update Cache. Update cache only brings in new data when the path has changed. Another difference is that if the path and library names do not change, Replace Cache reloads the part definition into the design. However, if Update Cache finds that the part name and the library names are the same, it does not bring in part changes.

⁶ Quotes taken from *OrCAD Capture User Guide*, Ch.17, [Project manager command reference](#).

5. PREPARING FOR PCB LAYOUT

When your design has been completely captured in schematic form, the next major step is to prepare it for the PCB layout phase. This involves several tasks: annotating and checking reference designators; checking for errors and warning by formally running the design rule check algorithm (DRC); mapping part symbols to wanted physical packages, called footprints or decals; and finally, creating an error-free *netlist*.

ANNOTATION – The first step preparatory to the PCB layout phase is to *annotate* it. This is the process of assigning unique reference designators to all parts. You probably took great pains to insure the instance designators for each part agreed perfectly with the handout reference schematic. This was unnecessary – but you wouldn't know that unless you already knew how to draw schematics and annotate the entire design as a single event here at the end. Generally, we just leave the instance assignments Capture gives each part as we add them and don't worry about what they actually are until we're ready to proceed to the PCB Layout. Moreover, if you copied parts already present, some instances may be duplicated, whereas if you deleted any as you went along, there will be corresponding gaps in the sequence of numeric assignments. We will now annotate your project as an example of how this is done.



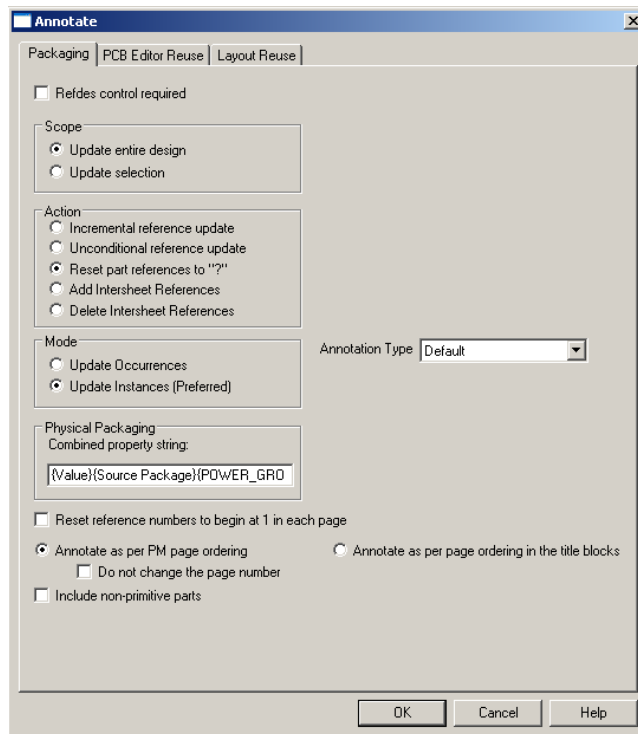
There are several ways to view all instances in a set of schematic pages. We could select all objects on each sheet using CTL-A, then observe properties on a per-page basis or, more quickly, by selecting the **Hierarchy** tab in the project manager. The example shown is from the handset reference design after properly annotating the schematic, so there are no part duplicates or missing numbers in the instances shown. (This may differ from your design.) Use this feature to quickly scan reference designators. Also, double-clicking any part will immediately bring up the page with that part high-lighted; a very nice way to find and check specific parts.



Since the annotation process repairs instance assignments, it isn't really necessary to keep track of them as you go along. There may be cases where we want certain parts to have certain assignments, like U1 or J1 for example, but this is easy to do and comprises part of the annotation process. We'll do this now for the tutorial schematic to show you a few of the annotation tools. In the file view of the project manager, highlight the design level virtual folder; this is the one with the *.dsn name: EE174tutorial.dsn. This will make the appropriate context-

sensitive menus selections in the Tool menu visible and active. From the main menu choose **Tools** → **Annotate** to bring up the tabbed annotation dialog, or select the icon shown (there is no hot-key available for this action).

Begin by resetting all instance references to “?” as shown on the Packaging dialog and select **OK**.



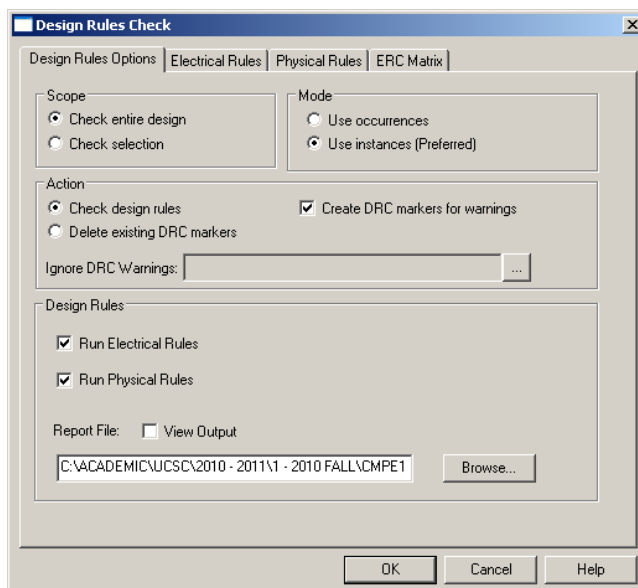
Check your schematic and observe that all references have been reset, then bring up the annotation dialog again. This time change only the action field by specifying **Unconditional reference update** and select **OK**. New instances will then be assigned to every part. Most likely, the 74HC00 will be incorrect, having the internal references wrong (the A, B, C, D series), and possibly these may also be assigned to two or more chips. Typically, homogeneous parts usually require subsequent manual editing.

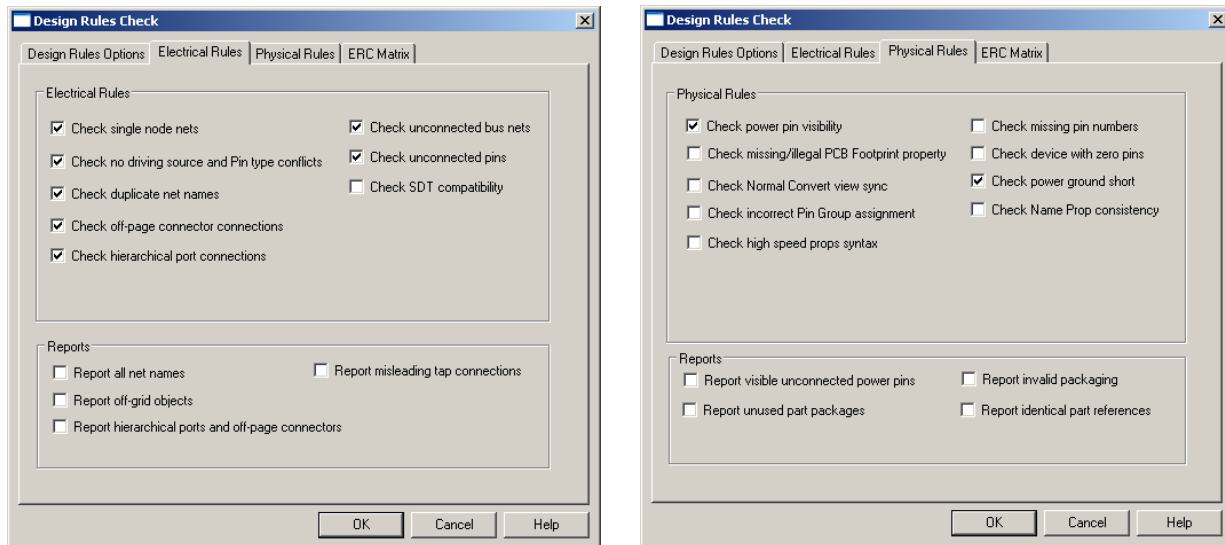
Go through the design and manually change reference designators to precisely agree with those indicated on tutorial schematic pages. This step is formally unnecessary, but since specific parts need to agree with their PCB placement in the next laboratory, it makes things much easier to have everything agree with my schematics.

DRC – The design rule check phase is the last thing that needs to be done to insure the integrity of our schematic-based design.

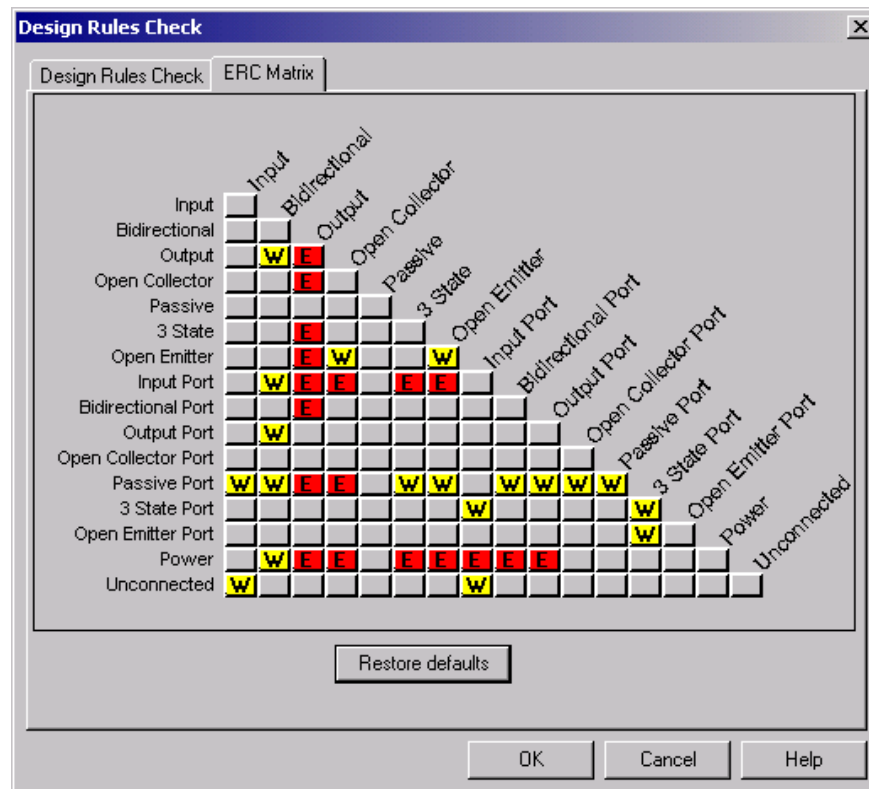


Again, first highlight the *.dsn virtual folder containing your schematic and either select the icon shown or the menu sequence, **Tools** → **Design Rules Check**, to bring up the Design Rules Check tabbed dialog. You will find it well worth the effort to learn to use the features available here. The **Design Rules Check** tab should be active by default. We often know exactly what we want to check and thus will usually choose a subset of the available Report options spread between electrical and physical Rules. Since we want to see DRC markers (little green donuts), check this. We will view the report via the session log, so leave the **View Output** option unchecked as well. Check both the Electrical and Physical Rules, since we will choose particular subsets of these.








Beginning with SPB 16.5, Cadence split the older single set of design rules into two new tabs: *Electrical Rules* and *Physical Rules*. These are shown side-by-side above. Additional rules were also added to each of these, particularly on the physical side. The Electrical Rules are shown at left; the Physical Rules on the right. For now select everything as shown on these two panels. The Physical Rules apply principally to the PCB context and less to the graphic schematic side here in Capture. Hence, not many items should be checked at this time. As you learn to use these, check them and run the DRC as appropriate. For example, we haven't specified any PCB footprints yet, so checking this would result in many errors. This particular error was missing in earlier versions and often had to be discovered only after going through the process of creating a final set of netlist files and reading them in Allegro PCB. Other physical errors won't be intelligible until you've gone through the whole process and gained a better understanding of how the whole thing works.



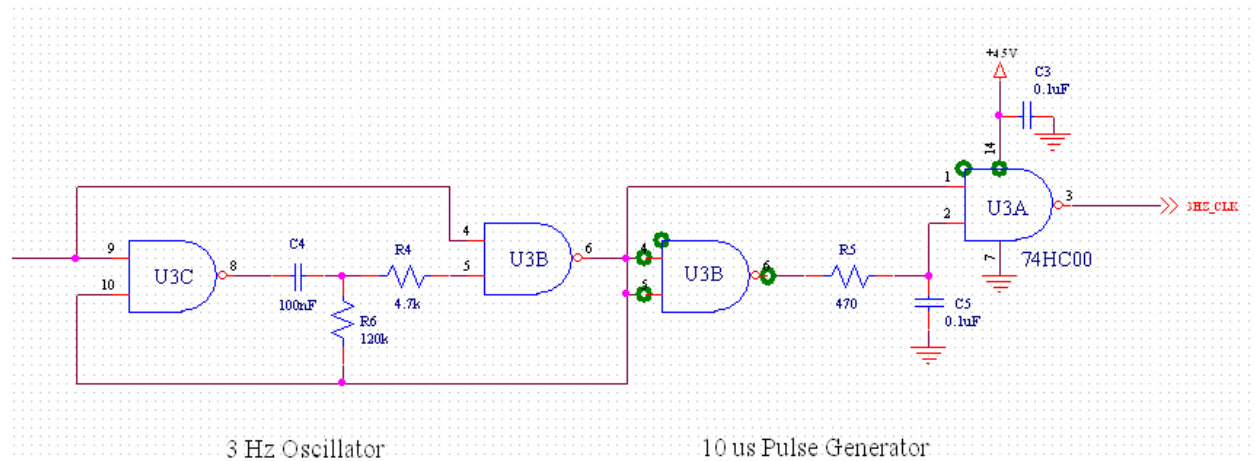
Finally, select the **ERC Matrix** tab. Shown is the resulting **Electrical Rules Check** matrix. It provides a way to fully specify what we want the DRC to check and how all net connections should be reported, as follows:

-  Report this type of connection as an **Error**.
-  Report this type of connection as a non-fatal **Warning**.
-  This type of connection is **OK**, don't report it at all.

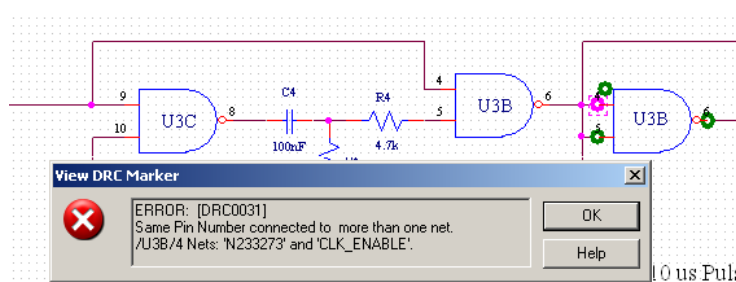
Capture uses the *pin type property* associated with each pin when carrying out the ERC. You can now appreciate why its important to get the pin type correctly specified when a part is first created. For example, wiring two input pins together is quite alright and does not constitute an electrical error; thus it's specified as "don't report". On the other hand, two outputs tied together can cause serious electrical problems and is appropriately flagged as an error. Warnings don't necessarily cause electrical problems, but are flagged simply to raise an alert; did we really want to this, or is it an oversight and hence an "error"? For example, a bidirectional pin (*i.e.* one capable of being either an input, output or floating) tied to a power net, like VCC or GND. We may have overlooked this or done it on purpose, knowing full-well the bidirectional pin will always be configured as an input. Therefore, all reported warnings need to be looked at to insure we didn't make a dumb mistake so they can be fixed or disregarded accordingly; that's why they're warnings and not errors. At this point press the **Restore Defaults** button to make sure the matrix has the correct default settings. And **DON'T MEDDLE WITH THESE SETTINGS** until you possess some engineering design experience and fully understand the implications of what you're doing. Capture will not allow you to go on to the next step of creating a netlist until ALL errors are cleared (but not warnings); again, **DO NOT** change these flags from their default settings in an effort to get a netlist created.

Now go ahead and run the DRC by pressing **OK**. The first thing you should notice is a new file appearing in the virtual Outputs folder in the Program Manager with a ***.drc** extension. This is a real file placed in the same physical folder containing your ***.dsn** file that can be opened by double-clicking the name in Capture or actually navigating to it via the usual windows methods outside of Capture. It is created the first time the DRC is run and overwritten thereafter. The same information is also appended to the **Session Log**. I like to look at the session log rather than open the ***.drc** file, but the choice is up to you. Here, we checked the View Output option, so the file is opened in the default text editor for your immediate perusal. You should now look at the DRC summary in both places: your schematic and the session log or ***.drc** file.

The schematic will display errors and warnings alike as “little green doughnuts” flagged next to offending points of interest. Double-clicking these will produce pop-up text boxes classifying the problem with succinct sentences stating the nature of the difficulty. The session log or *.drc file will also contain useful written summaries because of the nice way problems are classified. You should now look at your schematic and then correlate any errors/warnings noted with their respective descriptions in the session log or *.drc file.



Shown above is part of my schematic with a bunch of doughnuts. Your design may or may not show all of these (or even some different ones). The only flag that should definitely be on your schematic is the one at pin-14 of U3A. The homogeneous HC00 part is a good example showing us a very common set of flags that are often difficult for



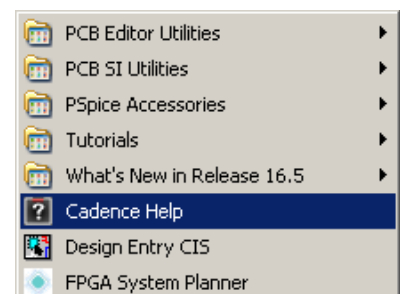
beginners to correct. Double-click on any of the flags about U3B. The resulting pop-up text associated with pins 4, 5 and 6 is identical. This error, DRC0031, tells us that net “N233273” is shorted to another net, “CLK_ENABLE”. The reason should be obvious by inspection: I have two instances of U3B! This is easy to fix, but suppose it isn’t quite so obvious, and we need a little more elaboration on this particular DRC error?

Pressing Help (below the OK button) is useless, since it merely makes a vacuous statement about the obvious (try it, and you’ll see what I mean). So *where* do we go to get some intelligent help?

TIP: Cadence help files are mostly a byzantine accretion of largely uncorrelated resources – basically, a mess. Many of them *are* useful if one only knows what to look for and where to find them - with nearly all of them embedded somewhere in the ...**(SPB16.6\docs)** set of subfolders (a total of about seventeen thousand files...). These folders appear to have their genesis in the days when file and directory names were limited to 8 bytes – hence many are cryptically not self-documenting, so it’s worth the effort to figure out the path to things that interest you.

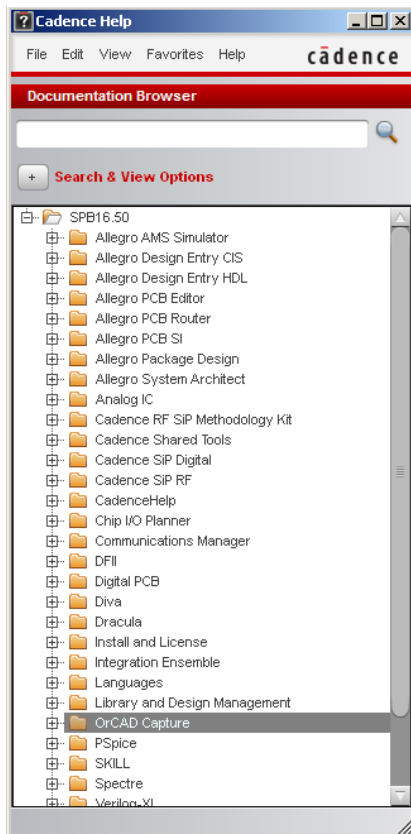
These folders can be accessed two different ways:

1) Begin from the main menu tree by selecting the “Cadence Help” option, as shown to the right. Cadence proceeds to load their separate shell application *Documentation Hierarchy* engine. Check “By Product” in the *View Documents* dialog box and then close the space-wasting *Search & View* interface by clicking the *Search & View Options* button (“--” symbol). From the Products level (next screen shot following), choose the tool you want to explore.



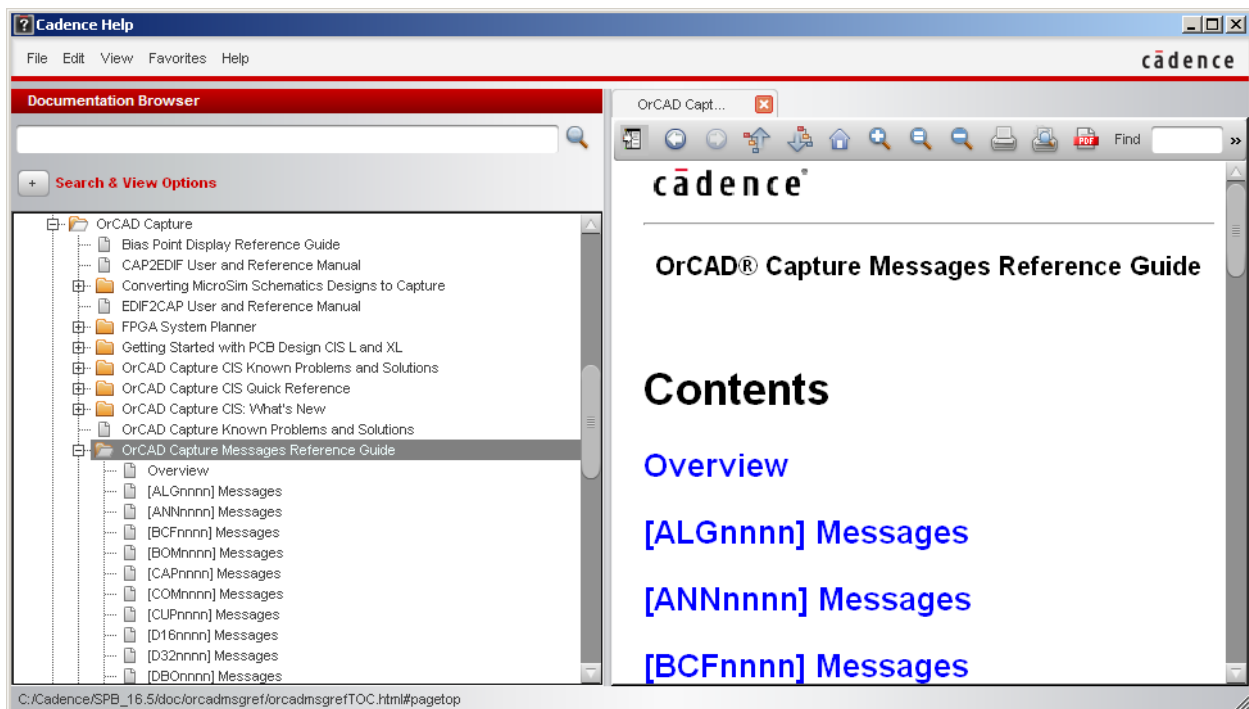
2) Another less obvious way is to boot the shell engine from within the active tool program itself, in this case Capture by choosing menu sequence **Help** → **Allegro Design Entry CIS** → **Capture Help**. From the vacuous

dialog *Cadence Help*, choose **View** → **Show Navigation**. This will load the same Documentation Hierarchy engine.



For example, I used the second way where I then selected *OrCAD Capture* as the tool, or “product”, to investigate; this causes a set of “manuals” to appear (see the sequential screen shots left and below). Typing search strings in the “Search manual” field usually results in substantively useful hits. Rather than do this (but you should try typing in *DRC0031* to see what happens), I just show the document we want: *OrCAD Capture Messages Reference Guide*. Selecting this “manual” accesses an HTML file somewhere in the doc tree noted earlier that serves as a wrapper for many dependent HTML files, or one possibly present PDF file found within the same folder. If a PDF version exists, it is always easier to use and search through than the HTML version, but not all topics have PDF forms. If it does, the topmost menu bar will show an active icon that can be pressed to bring up the much more useful PDF version; if this is greyed out, there’s no PDF available (see screen shot below). When this PDF option is selected, we observe in the Acrobat plug-in the source file name is *orcadmsgref.pdf*.

TIP: Generally, I have found the HTML versions don’t usually lend themselves to broad searches, since their architecture is organized as a set of discrete files, and only a file actually active can be searched (very annoying). The PDF, by contrast, is usually a single monolithic file that does lend itself well to broad string searches and should always be used if available.



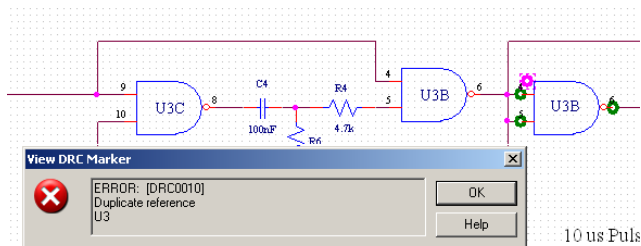
Loading the PDF version, searching for “DRC0031”, results in the following entry:

[DRC0031] Same Pin Number connected to more than one net.

Design Rule Check detected multiple pins with the same pin number connected to different nets instead of a single net. In a package there can be multiple logical pins having the same number as they physically represent a single pin only.

Note: To catch DRC0031 errors in your design, select the Check SDT Compatibility check box in the Design Rule Check dialog box.

Clearly these amplifying remarks make more sense than the terse single-sentence report in the **View DRC Marker** dialog box. Most of the time this will help clarify unfamiliar errors and warning, at least providing more clues to the solution. Changing one of the two instances of U3B fixes this.

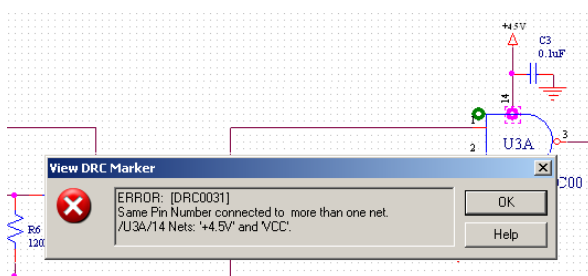


Had we selected the doughnut associated with the symbol body rather than its' pins, DRC0010 would have appeared instead. This one is pretty obvious but, again, suppose we need a little more elaboration. The corresponding entry found in the PDF version is shown following:

[DRC0010] Duplicate reference

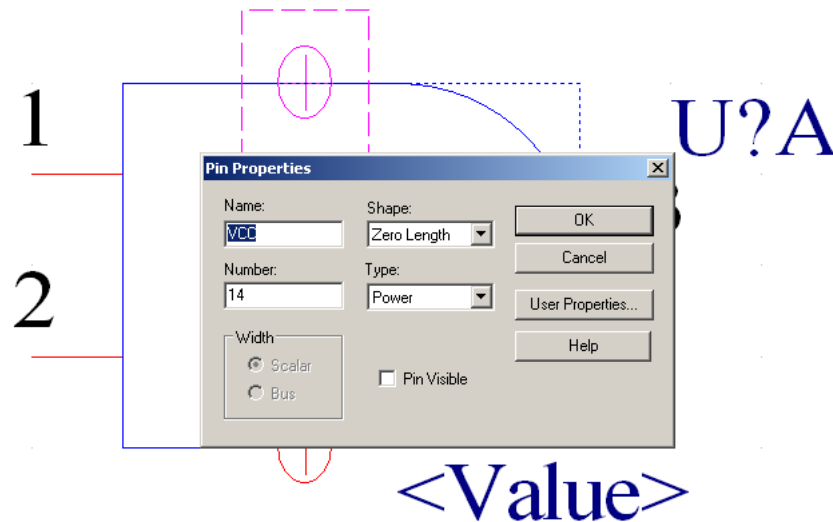
Design Rules Check encountered a duplicate reference. For example, this message occurs when two parts have a reference value of U1A. Increment one of the parts to a different member in the package (such as B or C), or change it to a different package (such as U2). Alternately, use Update Part References to first reset, and then update the part references.

These errors were easy to resolve, but the remaining two are not.



Choose the doughnut on U3A, pin-14 (double-click on it). This particular error message, DRC00031, tells us what the real problem is: we have somehow shorted two nets, “+4.5V” and “VCC”. The “+4.5V” net makes sense since this is what we manually named the net connected to pin-14 by the power connection symbol (the +4.5V arrow connected to U3A pin-14), but where did the “VCC” net name come from? The answer is that *it's the default name assigned to the part's pin*, and, since the pin type is also “Power”, *the pin name additionally specifies the invisible*

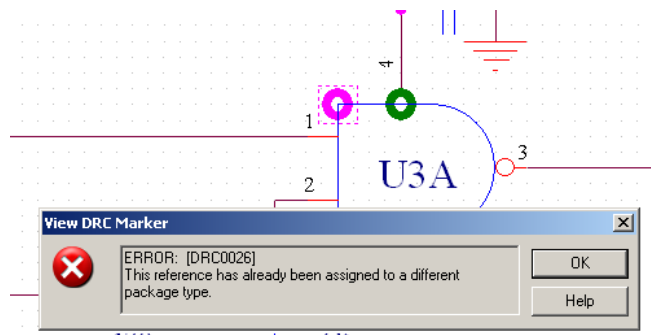
net it will automatically be connected to when the pin is optionally set to be invisible. Remember, this is done when any part is created. Note this is not the case with other pin types, like input, output, passive etc. (go back to the parts editor and note that only power pins can be optionally made invisible). Although these other pin types have names, they can never also serve as net names; *only power pins have this characteristic*. Select any of these four homogeneous Nand gates, right-click to select **Edit Part**, then double-click pin-14 to view **Pin Properties** and note the pin name is indeed “VCC”, so its intended to be connected to net “VCC”, and we've connected it to net “+4.5V” instead.



Fix this by either changing the pin name here in the parts editor to “+4.5V”, or correct the net name assigned U4A pin-14 by re-naming it “VCC” (remember to *Update All* so we modify *all four* instances in this homogeneous part). Since the net out of the LM317 power regulator chip has already been assigned net alias “+4.5V”, and this is really what we want it to be, we should rename the pin to agree with this alias. Remember we are working with the part in the cache, so making local edits like this is alright – it has no effect on the master library part. Generally, when selecting parts from Capture’s native libraries, any power pins will have been named something generic, like VCC, VEE, VSS, GND etc., so we often have to make this kind of local edit to specific cached instances.

You may also be wondering, in the first place, why Capture links the pin name with external net names only for power pins. As already noted, unlike other types, power pins can deliberately be made invisible. This practice developed historically to make schematics easier to read by implicitly acknowledging the obvious necessity and presence of these otherwise invisible power pins. This means real nets named “VCC” and “GND”, for example, must physically appear somewhere on the schematic for these invisible pins to connect to. This practice is most prevalent with digital designs. Why, then, would we ever *want* to show them? The answer is that it draws attention to needed self-documenting features, like the spatial placement of particular parts, such as the bypass capacitor meant to be connected close to U3A-14; the design *purpose* of the capacitor is to de-couple noise on the power supply bus and hence must be physically close to the chip to do this. Where would we show this if the positive power pin wasn’t made visible? Typically, all such bypass capacitors would then be grouped together somewhere on the schematic – usually at the bottom of a page, wired in parallel, and shown connected to VCC and GND. Their *design* purpose is lost and, unless text is added to qualify their use, like which chips they’re meant to bypass, the later PCB engineer probably won’t place them correctly.

Suppose, now, that I proceed to change the pin name property to “+4.5V”, choose *UpdateAll* and re-run the DRC only to find the two errors still appear! And worse, the same shorted net message comes up for the doughnut on U3A-14, but checking with the part editor, we find the name *is* “+4.5V”. How could this still be shorted to “VCC”? Understanding the answer covers a host of related warnings and errors - all caused by a single problem revealed by the last doughnut:



Further investigation results in the following elaboration:

[DRC0026] This reference has already been assigned to a different package type

The part's reference (such as R or U1) has been assigned to a different package type. For example, this warning occurs when a 74LS08 has been assigned a part reference U1B, but a 74LS32 part has been assigned a part reference U1A. Change the part's reference to an appropriate value, or use *Update Part References* to first reset, and then update the part references.

This warning also occurs when one part in a multi-part package is edited in place on the schematic. For example, this warning occurs when a part with part reference of U1D has been edited in place in the schematic, but U1A, U1B, and U1C have not. The edited part has lost its connection with the original library part. However, the other parts in the package have not lost this connection. Either use *Update Part References* to reset the edited part, or update the other parts to match the edited part.

This warning also occurs when one part in a multi-part package is edited and it results in an extra entry of that library part in the design cache. Either use *Update Part References* to reset the edited part, or remove the extra entry in the design cache.

The last paragraph is the key to what's going on. At some earlier point, I made an edit to this particular instance, U3A, and when I saved it, I wrongly chose *Update Current* (see pg. 35). This caused a new package to appear in the cache having some “_nn” suffix different from the one used for the other three instances. We can see this in the cache, or by carefully comparing the four homogeneous instances in the property spreadsheet, shown below:

Property Editor				
New Row... Apply Display... Delete Property Filter by: < Current properties > Help				
74HC00_18.Normal				
	A	B	C	D
Color	Default	Default	Default	Default
Designator	B	D	A	C
Graphic	74HC00.Normal	74HC00.Normal	74HC00_18.Normal	74HC00.Normal
ID				
Implementation				
Implementation Path				
Implementation Type	<none>	<none>	<none>	<none>
Location X-Coordinate	655	755	925	455
Location Y-Coordinate	590	600	560	600
Name	INS14174467	INS14174595	INS14174723	INS14174339
Part Reference	U3B	U3D	U3A	U3C
PCB Footprint				
Power Pins Visible	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Primitive	DEFAULT	DEFAULT	DEFAULT	DEFAULT
Reference	U3	U3	U3	U3
Source Library	C:\ACADEMIC\CUICSC\20	C:\ACADEMIC\CUICSC\20	C:\ACADEMIC\CUICSC\20	C:\ACADEMIC\CUICSC\20
Source Package	74HC00	74HC00	74HC00_18	74HC00
Source Part	74HC00.Normal	74HC00.Normal	74HC00_18.Normal	74HC00.Normal
Value	74HC00	74HC00	74HC00	74HC00

Notice the four parts have two different graphic representations: “74HC00.Normal” and “74HC00_18.Normal”. These refer to the two different cached versions that must otherwise be the same. Resetting the part is the easiest way to fix this problem. This is done by first deleting it on the schematic, then copying and pasting one of the existing three instances and re-modifying it as necessary but remembering to *Update All* when making any of these changes in the part editor.

Should you get any warnings, since they're not errors they don't necessarily have to be fixed, they must be understood. Warnings are for us. Capture notes a *potential* problem and want us to be sure we intended this. For example, a bi-directional pin type connected to an output only pin type will flag a warning, since the bi-directional pin could potentially be an output. Beginners tend to want to resolve warnings by eliminating them, but this belies their purpose.

Once you've fixed all errors and *deciphered and understood all warnings*, just turn them off while subsequently running the DRC to make the schematic less cluttered with little green donuts. Don't proceed to the next section until all errors have been cleared.

SCHEMATIC SYMBOLS AND FOOTPRINTS – Before compiling your project and proceeding on to the backend PCB phase in Allegro, every schematic part symbol must first be mapped to its corresponding **PCB footprint** (also known historically as a **decal**) and recorded using Capture's property editor. Cadence calls these and other chip-level as well, "packages", but we'll stick with the more industry-standard "footprint". Footprints are basically descriptions that specify physical aspects of a real part. Just as Capture references schematic symbols in *.OLB library files, Allegro similarly references footprint symbols, but as a collection of files always found within one principal folder: *symbols*. This directory is found in your **Lib** folder. Designs requiring signal integrity evaluations also use the information found in symbols and may include another folder called *devices*.

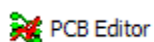
Note: The footprint symbols we will work with are generic. It is possible, however, to create a family of footprints, all being identical in appearance, but having different electrical parameters, for complete device families. Cadence provides a way to do this, and places the additional information in another set of files found in a parallel folder to the footprint symbols folder: **Lib\devices**. Thus, the device folder contains editable text files for families of actual devices having the same footprint but different parameters. We won't use this feature, so I did not add it to our folder tree. You can peruse an example by looking at the folders and files in the default installation at ...\\SPB_16.6\\share\\pcb\\pcb_lib. This folder contains both folders, symbols and devices.

At this point, the task now is to find and map correct footprints for each schematic part. We'll do this by using another tool, Allegro PCB Design, the same tool we'll be using in the next lab, but customized for the specific context of working directly with footprints. Here, footprints can be reviewed, copied, modified or created. We will use it now to review available footprints to obtain their names so we can map them in Capture.

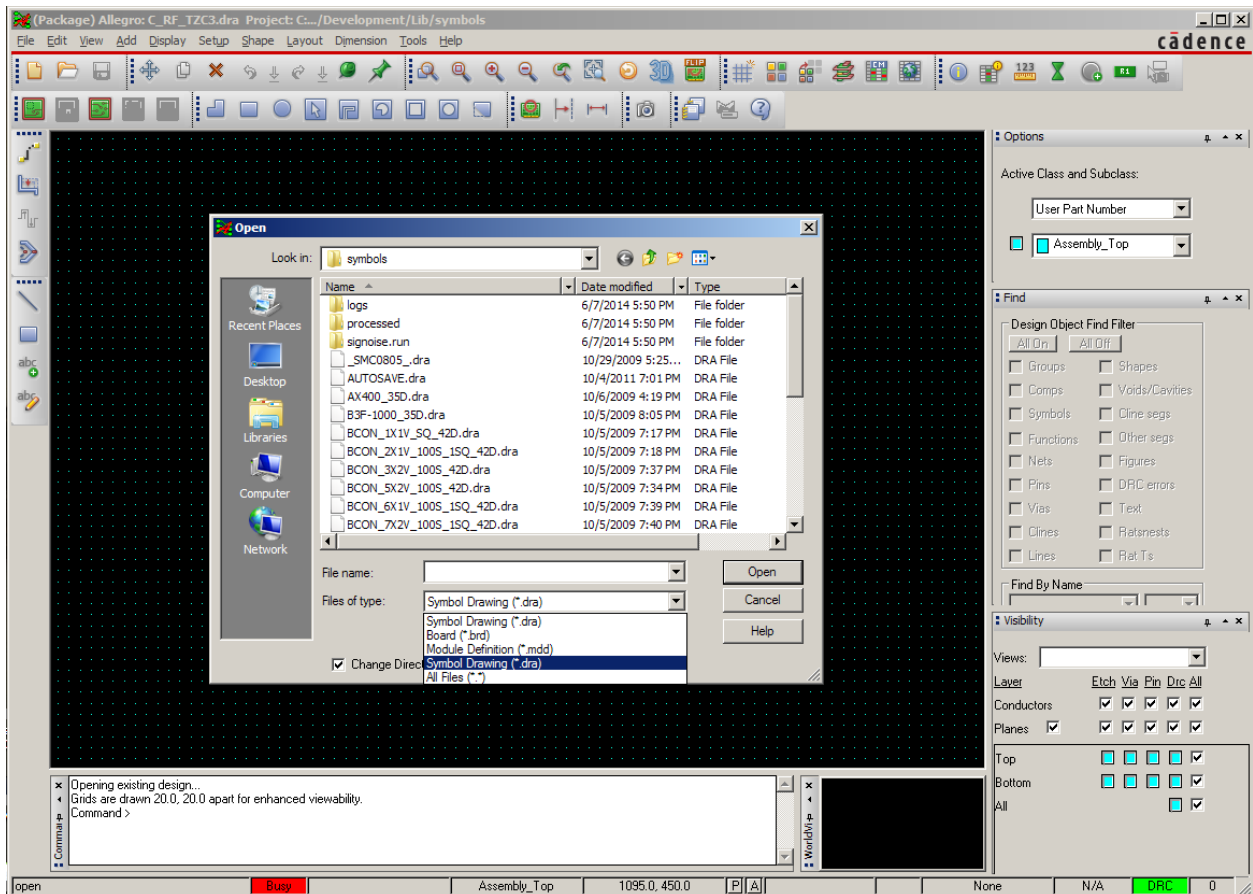
Each footprint is stored as a group of individual files having the same names but with different extensions; some of these are editable ascii files, others are not. Some of these files in the symbols folder may not be present for particular parts, since this depends on the kind of footprint it is:


...Lib\\symbols\\	
*.dra	Master drawing files containing, as a superset, one or more of the following symbols:
*.bsm	Mechanical symbols; non-electrical like mounting holes, board outlines etc.
*.psm	Component footprint symbols.
*.osm	Format symbols.
*.fsm	Flash symbols.
*.ssm	Custom pad shapes.
padname.pad	padstack definitions referenced by relevant symbol files above.

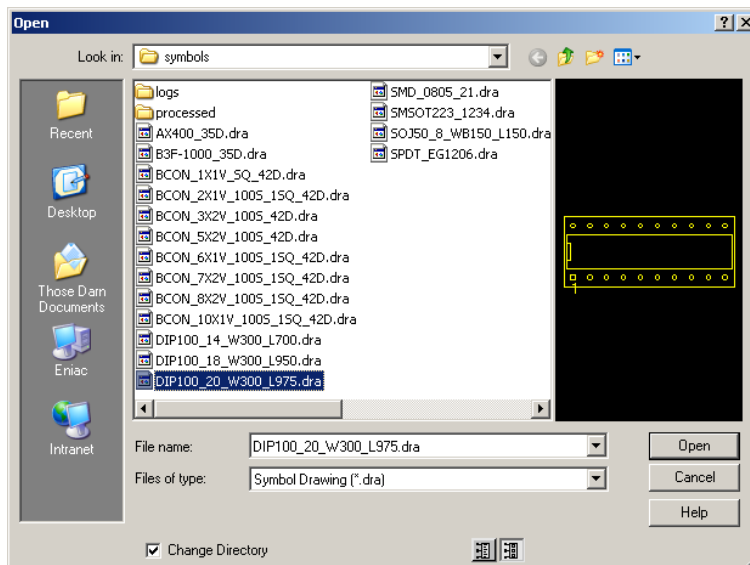
These files will be discussed in more complete detail in a later laboratory (lab 3), but only after building a suitable conceptual foundation from this and particularly lab-2.





Leaving Capture open but minimized, open a new instance of Allegro PCB Designer from the master window's start menu (entry and icon are shown left). Navigate to your own footprint symbols library and, by inspection, choose appropriate footprints for all parts in your design. Do this by following menu sequence **File** → **Open** and navigate to your ...lib\\symbols folder. Be sure the file filter is set for "Symbols Drawing(*.dra)". You should see something like the following screen shot:



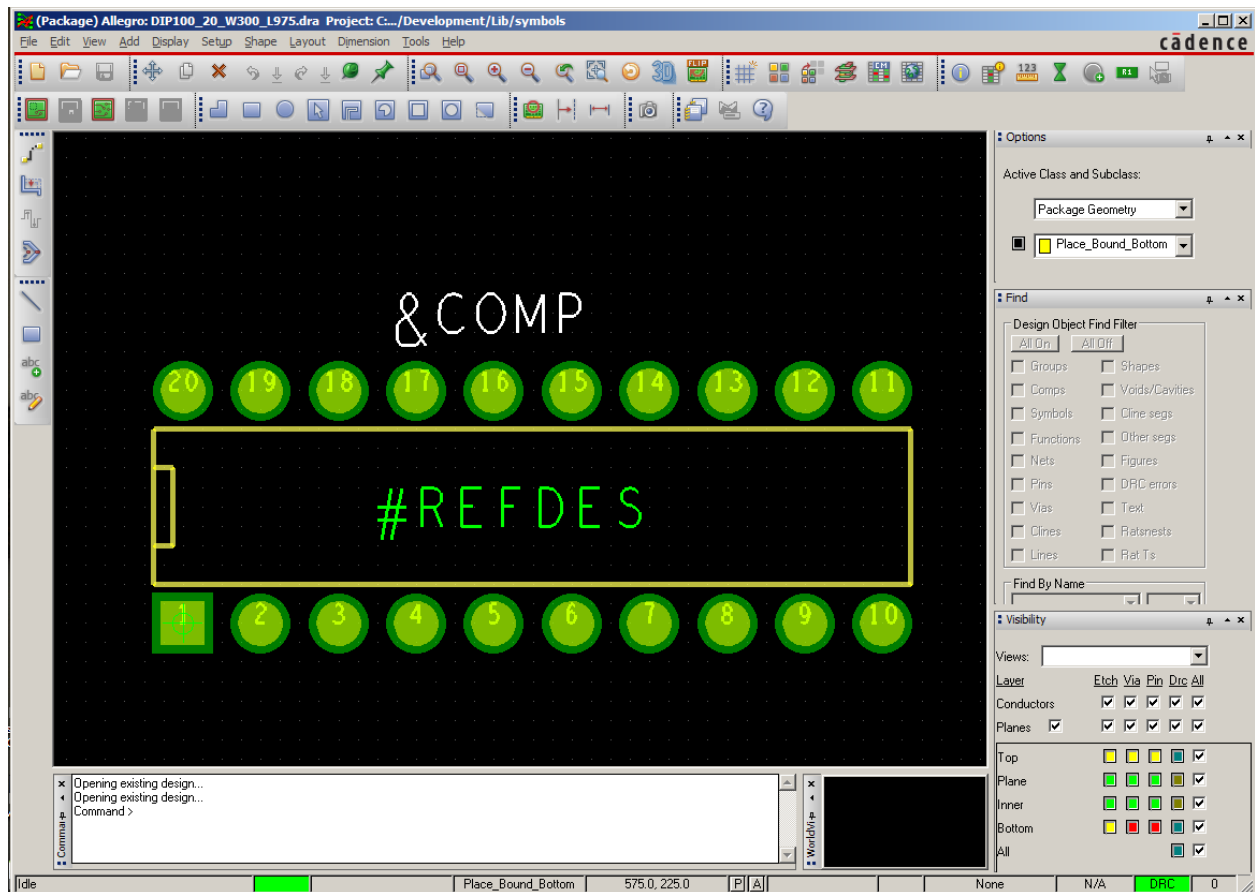
Since we haven't seen or discussed this program's user-interface (UI) before, I've included the whole screen shot so I can give you a brief overview. Be sure the top line reads *(Package) Allegro: ...* The remaining string tells us what part is currently in the editor and its source library. Allegro PCB Editor enters "package" mode when the specified file filter type is "Symbol Drawing (*.dra)". This is nearly the same window you will see in the next lab, when we use Allegro to layout the PCB when a "*.brd" file is specified. The three panels along the right edge of the screen, :Options, :Find and :Visibility present a dynamic set of context-sensitive interactive controls. You will learn how to use these in the next laboratory (lab-2). In the screens following, I have "minimized" these to flyout mode by pressing the thumbtack icon  on each; do the same now if they're not already "tacked" to the side.



For now, we want to concentrate only on the task of mapping footprints. Notice the two preview icons at the bottom of the dialog box:  . When pressed, both cause the same child viewer within the dialog to appear that is meant to provide some amplifying information about a selected part. The left icon will show textual information, the right icon a simple picture of what the basic footprint looks like.

For example, I have set the graphic preview mode and selected (but not opened) footprint **dip100_20_w300_1975.dra**. Together the thumbnail graphic and filename quickly tell us this footprint is for is a 20pin dual inline package (DIP) having pins spaced 100mil apart (0.100 inch); the two rows are 300mils wide; the IC's body is 975mils long. Note numeric

information is gleaned from the part's name and will typically follow some naming convention. Choosing text mode (left icon) is usually less informative (try it). Preview the footprint, then open it for editing. You will see the following:

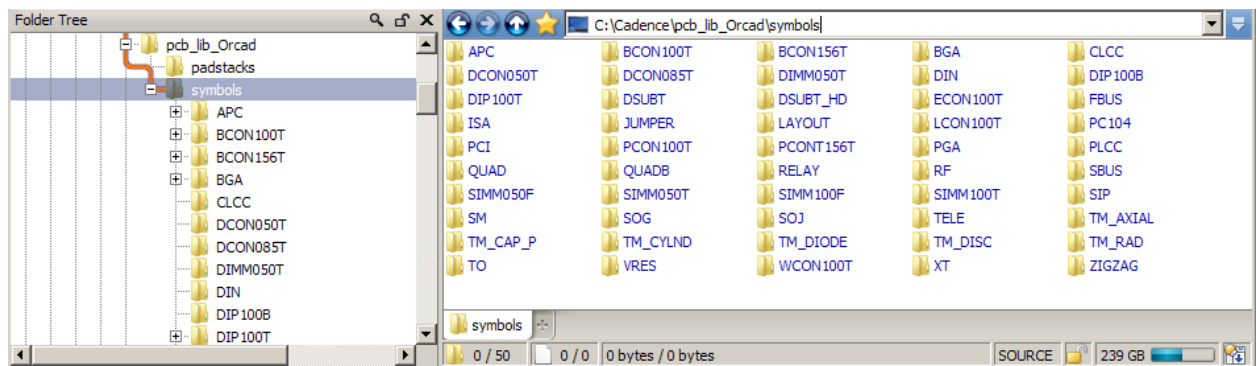


At this point we could make modifications to this part or copy it to another library. Observe that “#REFDES” and “&COMP” are placeholder fields that will eventually map to specific property strings after the footprint is bound to a particular schematic part and mapped over in the netlist files discussed later.

MAKING NEW PARTS – Typically, some particular custom footprints won't be in Allegro's native symbols' library and would otherwise have to be created by you. If you had to make them, this is the place to do it, here in the package editor. The workspace you see is equivalent to the PCB workspace you'll soon be getting experience with in Allegro (lab-2), and as already noted the packager is used specifically to work with footprints. We will delay the daunting task of learning how to make footprints here from scratch until you've had an opportunity to see how footprints and padstacks are actually used in the process of designing a PCB. At this point, we just want to identify the correct footprints and get them mapped correctly over into Capture.

FINDING FOOTPRINTS – PCB symbol footprints can be found in two places:

1. Allegro's native footprints are found by default in ...\\SPB_16.6\\share\\pcb\\pcb_lib\\symbols. This folder contains over 1100 files, representing hundreds of footprints. Their naming convention is not very informative, but some of them can be viewed for quick perusal in *PCB Editor User Guide: Defining and Developing Libraries*, sec. 8, pgs. 101 – 220⁷.
2. A special set of library folders I exported from the old Orcad Layout-Plus group that is much better organized and useful overall is available as a single zipped folder in C:\\Cadence\\pcb_lib_Orcad.zip. It is also on our website.. You should copy it into your profile and unzip it into the ...\\Lib\\pcb_lib_Orcad\\ folder you created earlier (noted on page 2). In the symbols' subfolder you will see 53 additional subfolders containing about 3000 footprints. Each of these subfolders contains a single Allegro *.brd file that can be opened to inspect the entire set of included footprints (this is covered in lab2) within that subfolder.

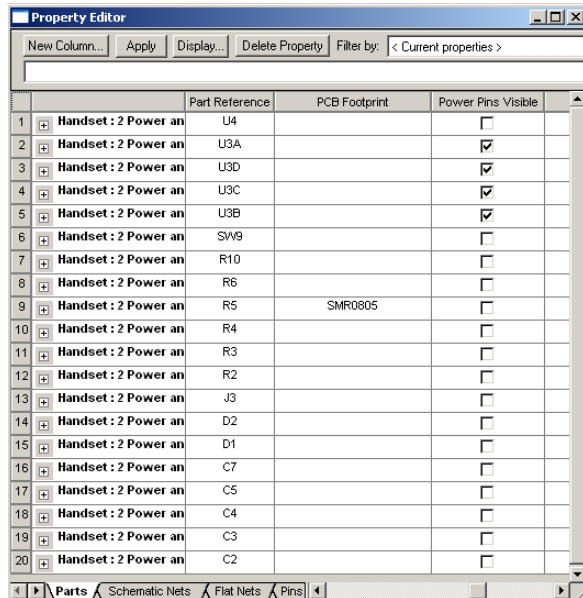


A brief but partial list of what these folders contain:

- **BCON100T:** Block connectors having 0.1" or 100mil spacing. These are typically used for "stake headers" and ribbon connection sockets and jumper blocks.
- **BCON156T:** Same as above except pin-to-pin spacings are 156 mils.
- **BGA:** Ball grid array footprints.
- **DIP100T:** Dual inline sockets having pin-to-pin spacings of 100 mils with through-hole vias; includes classic TTL sockets for chips like the 74HC00.
- **DIP100B:** Butt dips having rectangular pads spaced 100mils apart but meant for surface-mount soldering of otherwise through-hole parts like the 74HC00 note above. These chips would have their pins clipped before mounting.
- **SM:** Surface mount parts, like 0603, 0805, 1206 etc.
- **SOG:** Gull-wing SM parts.
- **SOJ:** J-mount SM parts.
- **TM_?:** These folders contain "Through-hole mount" discrete footprints for capacitors, round capacitors ("CYLND"), diodes; ceramic disc capacitors; radial-leaded devices.
- **TO:** Many variations of classed through-hole devices, like TO-220 etc.
- **VRES:** Variable resistors.

⁷ Accessed using the Documentation Hierarchy engine; see the Tip on pg. 41 for guidance.

MAPPING PCB FOOTPRINTS IN CAPTURE– Leaving the package editor just as it is, re-open Capture full-screen. The process I'm about to describe should be done for each schematic page (in this design there are only two), so begin with page 1. Select all the parts at once by typing CTL-A, right-click to choose **Edit Properties...** and bring



up the property editor. Pivot the fields if necessary and set the **Filter by:** for *<Current properties>* in the list box as shown. Choosing other filters will display a subset of properties specifically related to the task at hand making it easier to see things by removing those unused; for example, you could also have selected *Orcad-Layout* instead. I chose this one because it nicely arranged the three visible columns. Be sure to select the **Parts** tab at the bottom of the page.

We now want to specify the **PCB Footprint** properties for all parts missing them by going back-and-forth between Allegro Packager and Capture's property editor to select and enter the correct footprint names. I've shown one of them, for R5, having a surface mount 0805 footprint. Typically, we usually have to make one or more footprints along the way since they won't be in any of the native source libraries.

Since your reference designators may be different than the reference schematic, we will use the reference schematic provided with the lab as our common point of reference. Thus, you may need to cross-reference indicated parts to your newly annotated schematic. I will take you through the process for a few parts and let you finish the rest.

First, check that all the resistors are mapped to 0805 SMT (0.080 x 0.050 inch surface mount) footprints. This footprint can be found in the `pcb_lib_Orcad\SM` library noted above. After you find it, copy into your symbols library, then cut-and-paste the name from the File name string (without the `dra` extension) back into Capture for every resistor in your design.

All capacitors should also be mapped to 0805 SMT footprints as well. For reasons that will become clear later on, use the `SMC0805` entry in the `SM` library, which actually has the same identical footprint as the `SMR0805` entry above. Find and copy the `SMC0805` footprint over to your library then cut-and-paste the name as before, assigning it to every capacitor in your schematic using the open property page in Capture.

Find all the following footprints from among Layout's libraries and add them to your own working library, then assign them to their appropriate parts in Capture. Note, at this point, you may want to change any of your reference designators that don't agree with mine. Otherwise you'll need to carefully cross-reference your schematic with mine.

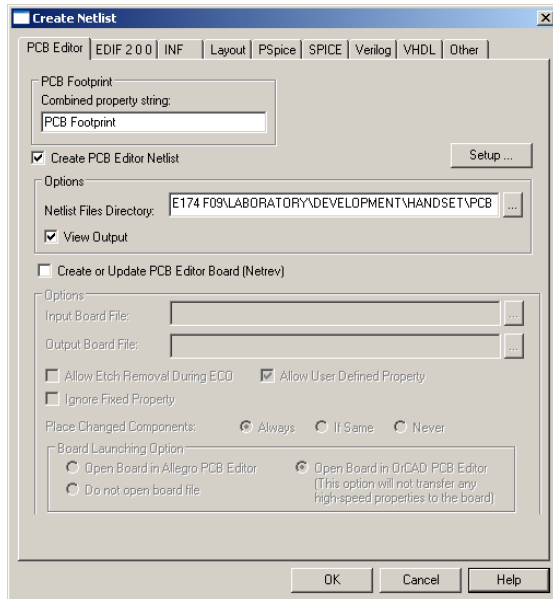
Part Reference	Value	PCB Footprint	Source Lib
C1	0.1uF	SMC0805	SM
C2	100nF	SMC0805	SM
C3	0.1uF	SMC0805	SM
C4	0.1uF	SMC0805	SM
C5	0.1uF	SMC0805	SM
C6	10nF	SMC0805	SM
D1	1N5818	AX400_35D	EE174TUTORIAL
D2	GREEN	SMD_0805_21	EE174TUTORIAL
D3	RED	SMD_0805_21	EE174TUTORIAL
J1	JUMPER8	BCON_8X2V_100S_1SQ_42D	EE174TUTORIAL
J2	CONN TRBLK 10	BCON_10X1V_100S_1SQ_42D	EE174TUTORIAL
J3	HEADER 2Pin	BCON_2X1V_100S_1SQ_42D	EE174TUTORIAL
J4	CONN TRBLK 10	BCON_10X1V_100S_1SQ_42D	EE174TUTORIAL
J5	JUMPER3	BCON_3X2V_100S_42D	EE174TUTORIAL
J6	HEADER 2Pin	BCON_2X1V_100S_1SQ_42D	EE174TUTORIAL
Q1-1	TPS1120	SOJ50_8_WB150_L150	EE174TUTORIAL
Q1-2	TPS1120	SOJ50_8_WB150_L150	EE174TUTORIAL
R1	1M	SMR0805	SM
R10	1k	SMR0805	SM
R11	1k	SMR0805	SM
R2	4.7k	SMR0805	SM
R3	470	SMR0805	SM
R4	120k	SMR0805	SM
R5	R	SMR0805	SM
R6	240	SMR0805	SM
R7	47k	SMR0805	SM
R8	47k	SMR0805	SM
R9	47k	SMR0805	SM
SW1	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW2	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW3	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW4	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW5	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW6	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW7	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW8	SW PUSHBUTTON	B3F-1000_38D	EE174TUTORIAL
SW9	SW_SL_SPDT	SPDT_EG1206	EE174TUTORIAL
U1	HT12E	DIP100_18_W300_L950	EE174TUTORIAL
U2	ATF16V8BQL	DIP100_20_W300_L975	EE174TUTORIAL
U3A	74HC00	DIP100_14_W300_L700	EE174TUTORIAL
U3B	74HC00	DIP100_14_W300_L700	EE174TUTORIAL
U3C	74HC00	DIP100_14_W300_L700	EE174TUTORIAL
U3D	74HC00	DIP100_14_W300_L700	EE174TUTORIAL
U4	LM317A	SMSOT223_1234	EE174TUTORIAL

THE SANITY CHECK!!— After finishing the PCB footprint mapping phase, bump the revision number to 0.9, save the project, make a hard-copy of the schematic and put it in your engineering notebook. Having version 0.9 in your engineering notes establishes an audit trail you can reference to figure out what went wrong later on. Now, using any notation you like, indicate *for every single part on the printed schematic* that you have checked it for correctness; this includes:

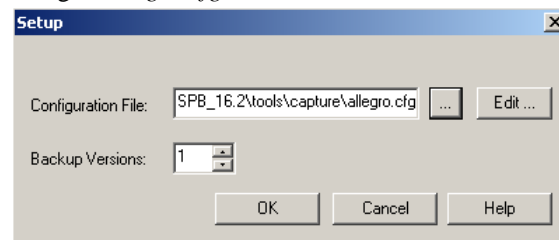
1. Checking for proper mechanical footprints and pinouts AGAINST the appropriate datasheets and mechanical drawings. A very common error is to get diode polarities reversed. For example, D1 and D2 are 0805 LED's. Be sure pins are properly defined (we used pin 1 and 2; some designs use Anode and Cathode – A and C instead. These are also in the SM library). So, there *may be errors* in the table above! Since you didn't have to create any footprints, only the pinouts might be wrong. An error that has occurred among students many times is choosing a surface mount footprint for a dual-inline package that “looked” enough like the mechanical drawing in the datasheet to appear correct, but is actually too wide or narrow and the pins don't end up aligning correctly on the finished PCB. Again, since you didn't have to create any footprints this go around, any such errors are due to prof. Petersen ☺.
2. Check pins of type power on relevant parts. In designs where these are hidden, it's very easy to forget to locally edit the pin names for the correct power net, especially when multiple nets are present, like +3.3V, VCC, +5.0V etc. This might apply to “grounds” as well, since some designs will have separate analog and digital grounds having different net names (remember the ground symbols with different net names discussed earlier).

It may seem odd, but don't advance the rev number to 1.0 thinking you've finished the schematic at this time. After all, you've got a netlist and are ready to proceed on to the backend phase. The reason is a good one. As you proceed to layout the PCB, there may be changes you want to make in the schematic that will only become apparent later. We commonly wait until the PCB phase to make final decisions regarding specific choices of internal instances, like whether to use the A, B, C or D instances in U3; re-arrangements can make actual wiring (or “routing” as we say) much easier. I'll remind you at the end of the next lab to bump the schematic to ver 1.0 after finishing the entire process.

CREATING THE NETLIST – When you finally have an error-free schematic with all PCB footprints properly defined and a fully checked version 0.9, proceed on to create the netlist! The netlist is just what the name implies – a complete listing of all parts, footprints and their interconnected nets. Capture allows us to create a variety of such netlists depending on the specific backend layout tool targeted. Since our workflow targets Allegro PCB Design, we will be creating a netlist to be read by that tool. Do this by bringing up the **Create Netlist** dialog using the icon shown or menu sequence **Tools** → **Create Netlist...**



Select the dialog tab for **PCB Editor** to specify we want a netlist created for the Allegro PCB Editor. Press **Setup...** to make sure we will be using the correct configuration file for Allegro, *allegro.cfg*:



Confirm the netlist will be created in your working project folder, *...\Handset\pcb*. Note that we could have checked *Create or Update PCB Editor Board (Netrev)*; this option automates the immediate creation of a new Allegro PCB board file, but I want to show you how this is done manually so you'll gain a better understanding of the general process. We

will, however, be using this option later when I show you how to reflect changes made in Capture to an existing PCB layout design; this process is called *forward annotation*. And, if you're especially curious, select **Help** to read about this window. For now, just press **OK** to create the new netlist file.

If the netlist was successfully created, three files will be written to your *...\Handset\pcb* folder:

1. **PSTCHIP.DAT**, a file containing a list of each component part present in your schematic.
2. **PSTXNET.DAT**, a file containing the actual “netlist”, *i.e.*, all named nets and their respective node attachments, net properties (bus, wire etc.).
3. **PSTXPRT.DAT**, a file containing the *expanded parts list* for all physical parts (footprints) mapped to each reference designator.

These filenames are a bit goofy, so I highlighted relevant sub-strings in each to help you better remember them. Knowing them by rote isn't really necessary, but seeing “chip” brings to mind componets, like resistors, IC's etc. Similarly for the other two files. These are all editable ASCII files and can easily be viewed or modified in any text editor. Since we checked *View Output*, you will be able to immediately see and edit their contents; for now, just peruse them to see what's inside and close each one.

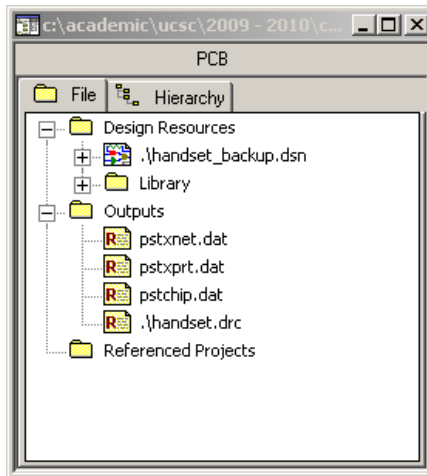
As it is, the process should have failed! Immediately consulting the session log tells us why:

```
#1 Error [ALG0044] Conflicting values of "Power Pin Visibility" found on different sections of U3: Handset, 2 Power and Oscillator (6.48, 3.92)
#2 Aborting Netlisting... Please correct the above errors and retry."
```

Error ALG0044 results from an annoying architectural bug in the design of this particular netlister, going from Capture to Allegro (other netlisters don't complain, and Capture's DRC didn't seem to mind). It occurs because we made one of the four homogeneous part's power pins visible, but didn't also make them visible for the others. This is silly, since the whole purpose of doing this was to explicitly indicate the power pins for the *single IC part package*, U3. Making them visible for all instances is redundant and now makes the schematic look like we've failed

to connect nets to the other visible, but apparently unconnected, power pins. You will have to go back and fix this before proceeding. Do this quickly by selecting just the four NAND gates (use the CTL key to select them) and setting their *Power Pins Visible* properties to all agree⁸. Re-running the DRC will now show three green donuts warning these gates have unconnected power pins. Suppress this particular warning by un-checking the *Report visible unconnected power pins* on the **Design Rules Check** dialog.⁹

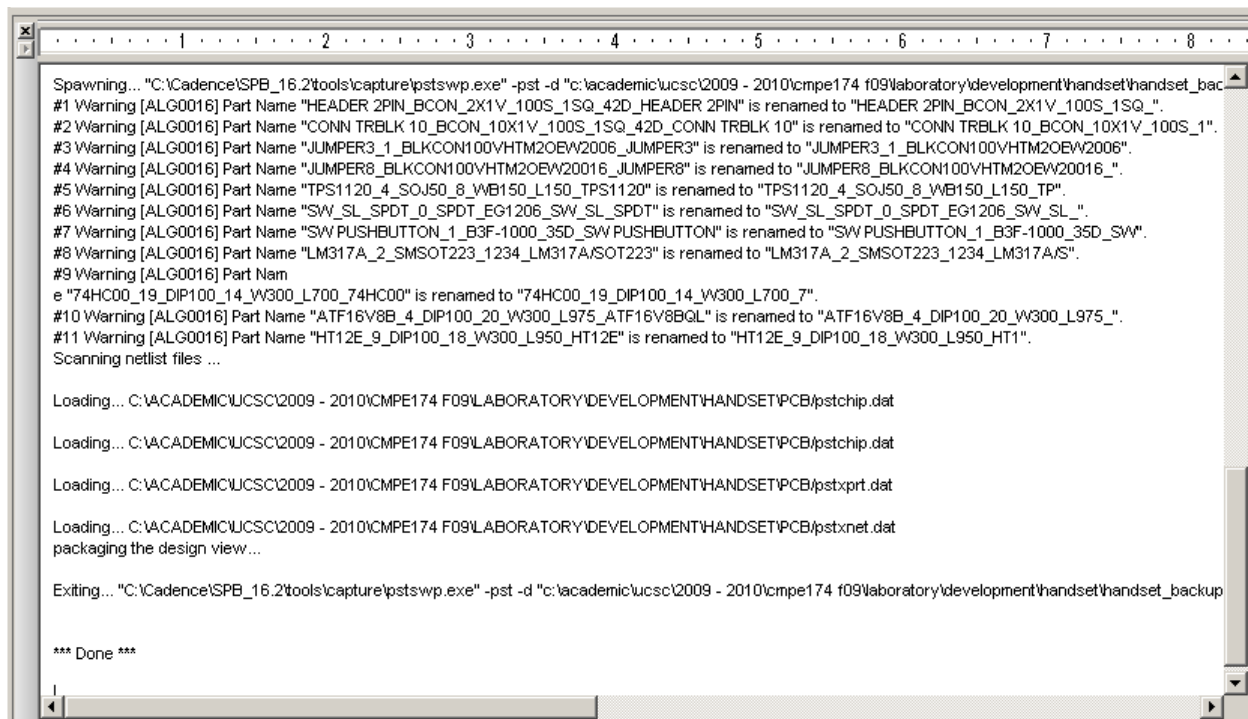
Rerun the netlist and, assuming you have no other errors, the three files will now appear listed in the virtual **Outputs** folder in the Project Manager along with the earlier created *.drc file as shown.



Selecting any of these with the mouse will bring them into a text editor for review or editing. This is another way to access them besides checking the *View Output* option.

If these files are generated, it merely means there were no errors, *but there could still be warnings*; you should always check the Session Log to be sure everything got translated without difficulties. The log will probably contain one or more ALG0016 warnings telling us that some part names are too long and the netlist is truncating them. This is another annoying archaic feature rooted somewhere in the distant history of the Allegro tool that limits part names to 32 bytes. Even though the footprint property strings we entered were not this long, the netlist appends the actual schematic part name (not the reference designator) to the that string before exporting the netlist, and it is this final string that is too long; don't worry about it. Leave it as is and go on, but note the names later appearing in the

backend tool (Allegro) will be truncated. An example of my session log is shown below.



⁸ See the Property Editor screen shot on pg. 46.

⁹ A screen shot of this window is shown at the bottom of pg. 39.

6.0 CONCLUSION

This concludes the first laboratory. Laboratory 2 will continue beginning with the three netlist files created in this lab. The following list addresses Capture-related topics that you might want to investigate on your own. Some of these we will address in this class, but the tutorials presented in this class should provide a firm enough foundation to address the remaining ones on your own.

1. How to export and import properties, add properties etc..
2. Creating and using the Bill of Materials.
3. Heterogeneous parts; instances and occurrences.
4. Special topics, such as PSPICE simulation and HDL flow.

Obtaining full credit for this lab will be depend on the following:

1. The completeness and quality of your engineering notes, consisting of the bound and loose-leaf notebooks.
2. Properly drawn and annotated engineering schematic capable of producing an error-free netlist.
3. Demonstrate competence and understanding by oral interview with prof. Petersen. Note this will be done interactively at a lab computer having your project files available.