University of California, Santa Cruz
Electrical Engineering Department

PCB EDA Tutorial
Laboratory 2, PCB Layout (PSB 17.2)

EE174, Introduction to EDA Tools
S.C. Petersen

Contents

1. Introduction and Objectives

This is the second laboratory in a series meant to comprehensively introduce the EDA workflow associated with board-level PCB design. These labs build on one another, so you should consider earlier labs as references to consult as necessary. *The PCB EDA Tool Chain* is an introductory essay meant to establish a conceptual foundation of understanding. This white paper should be reviewed as necessary to keep a clear system-level perspective in mind as you work through the many details.

The tutorial project introduced in the these labs involves two printed circuit boards that together make a hand-held remote control for transmitting telemetry commands to an autonomous robot. The heart of the electronic design is a PLA to interpret various pushed buttons that control a state machine which generates coded 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") to 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 has been included for completeness and reference.

The previous laboratory, *Laboratory- (Capture)* discussed front-end board-level project organization and dealt specifically with creating schematics using *Cadence Allegro Design Entry CIS* (we agreed this tool would be referred to simply as "**Capture**"). This second lab continues on with an introduction to the printed circuit board (PCB) layout phase, where we design the physical PCB's layout and generate output files necessary to producing a bare handset PCB. Details of the the actual PCB layout are included as Appendix-C. You might want to print this or otherwise make it available to refer to as you work through the layout phase. Note Appendix-C has been extracted as a separate file, **Lab-2_Appendix-C.pdf.**

Before You Begin.
You should have already created the following set of directories. I include them here as a convenient reference and reminder of their relevance. Refer to Lab-1 for details on their location and creation.

…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).

During the process of using Capture and Allegro, especially Allegro, additional irrelevant directories may also 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!". Remember that Appnotes, Datasheets and Components synchronize with material that may also be placed in your engineering notebook[1] under these same headings. Generally, these directories serve as source repositories for information you find important enough to then print.

> **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.

2. General Discussion

Designing printed circuits having discrete components is a two-step process: First, the design must be fully entered into a suitable engineering schematic that accurately includes all relevant electrical components and interconnection details. This front-end phase was completed in the previous lab that resulted in a set of compiled output files fully encapsulating the schematic design as a *netlist*. Each component created for use on the schematic side typically includes hierarchical information pertaining to mechanical facts necessary to actually laying out this part on a PCB. This includes such things as sockets, pad sizes, etc., and are typically called *decals* or *footprints*. I mentioned in the last lab Cadence calls these objects "packages", but that we would refer to them as footprints.

Do not proceed to start a new board unless the following has already been done:

- The schematic design has been created using Capture and all DRC warnings and errors have been resolved.
- A set of compiled files representing the complete netlist has been generated.

2.1 PCB Background

Printed circuit boards are designed using highly specialized mechanical drawing programs that allow designers to place various elements of a board's design on different logical and physical layers with each layer serving a specific purpose. This layering scheme allows us to easily separate things for manipulation and visibility. For example, consider a generic two-layer PCB. This board has a top copper layer (known historically as the *component-side*) and a bottom copper layer (on the back known historically as the *solder-side)* separated between an insulating material. When a board is made, we need to specify more than just copper pads and traces. Besides these two physical layers assigned to the top and bottom etched copper routing layers, logical layers, or "documentation" layers, are also defined for things like drill targets, solder masks, silk-screens, component placement, mechanical drawing of the board's outline, special fabrication instructions, etc. Defining parts normally involves several logical layers too. An example is the famous *via*. These are electrical circuit connections between physical layers (for example, top to bottom) and require additional information on several logical layers to completely specify them. This collection of information, called a *padstack,* is included in PCB part or footprint libraries as a single named object.

The phrase "*layer stackup*" is industry-speak for the complete physical description of a particular PCB. For example, a board having four overlaid copper layers would be referred to as a "4-layer stackup"; note that even though technically the dimensionally stable solid dielectric insulating layers between them are also physical layers, these are not included in the layer description. Cadence calls a board's complete stackup its *cross-section* that includes the insulating layers. This more dignified term is taken from mechanical engineering, where an object when viewed from the side is called a cross-sectional view and obviously shows everthing.

A typical set of layers for a simple 2-sided stackup might have a complete cross-section described as follows. Remember, any logical layers are mechanically registered, aligned or overlaid with each other, right along with the cardinal physical copper layers. Classifying a particular layer is less important than understanding what it does and how it's meant to be used.

1. **Top-side Copper.** This layer contains all etched physical traces (these are the actual wires or conductors), pads and "copper pours", large areas of solid copper.

[1] See ***The General Practice of Engineering Notes***, found on our website.

2. **Top-side solder-mask**. This logical layer defines the finished board areas to be coated with a thick enamel film, usually green (blue, pink, yellow, black and red too) applied over the freshly etched bare copper board. It is a painted layer meant to mask regions where we don't want solder applied to a board. Basically, it marks off areas where components or vias are actually soldered, along with areas we want "tinned" with a thin protective layer of solder. Manufacturing usually specifies application of this layer as *solder mask over bare copper (SMOBC)*.

3. **Top-side paste-mask**. Automatic assembly of PCB's is done by machinery. This layer defines areas specifically relevant to the mechanized application of fluxes and glues needed for automatic placement and soldering of surface mount components. This logical layer isn't needed for manually assembling a PCB.

4. **Top-side silk-screen**. This documentation layer is also known as the "legend" layer and has things like lettering, parts identification and outlines usually applied as bright white enamel. Technicians and assemblers can more easily identify components this way. Basically, any written information we want on the finished PCB is defined here. Traditionally this has been done by literally making a classic "silk-screen", laying it over the finished solder mask and rolling ink over the screen to apply the artwork.

These layers are all repeated for the bottom-side as well. Traditionally, the top-side of the board is sometimes called the *component-side* and the back or bottom-side the *solder-side*. These designations are historical and often don't reflect which side we actually place components on, but we generally begin by placing components on the top-side.

5. **Drill Targets**. This logical or documentation layer exclusively show the location and size of every drilled hole on the PCB. Holes meant to connect top and bottom-side traces are known as *vias* and are first drilled and then plated with copper that connects to copper pads on each side respectively. Vias are also known as *plated-through-holes* (PTH). Drilled holes that aren't meant to be vias will have no copper plating and are designated NPTH or NPH; these are usually for mechanical mounting.

6. **Assembly Drawing**. This logical or documentation layer never actually appears on a finished physical board. It typically has a fully dimensioned outline of the PCB, a table of drill sizes and number of associated holes, their type (NPH or PTH). Moreover, it may or or may not be sent on to the board house, but if it is, it may contain additional special fabrication instructions.

7. **Inner Layers**. These are etched physical layers sandwiched between the top and bottom-side physical layers, and, for mechanical stability reasons, are always added in pairs (prevents warpage). They have a variety of standard uses:

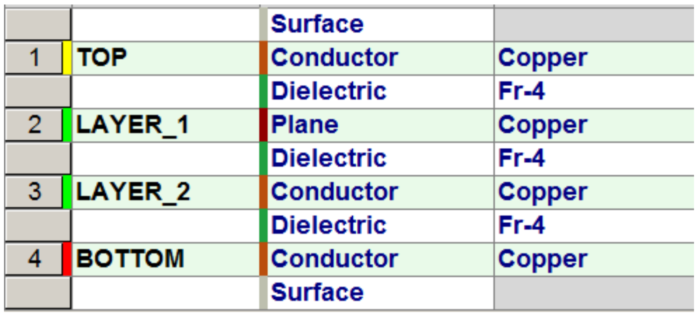
    - *Routing Layers*. Large designs having many nets typically need additional layers to sufficiently wire everything together given size restraints.
    - *Power Planes*. Routing power is often a problem. By making power traces large planes of copper on inner layers, electrical noise can be reduced and current-carrying capacity increased. EMI (electromagnetic interference) and EMC (electromagnetic compatibility) can also be dramatically improved.
    - *Transmission Lines.* High-frequency analog and high-speed digital designs often require nets implemented as transmission lines having controlled electrical lengths and characteristic impedances. Various kinds of transmission lines can be constructed using inner layers. These include single-ended and differential *stripline;* single-ended and differential *microstrip*. Although microstrip transmission lines can be constructed on a two-layer board without inner layers, routing density and realizable complexity is greatly reduced.

    The presence of inner layers significantly increases PCB manufacturing cost, and should only be used when really necessary.

8. **Special logical layers** depend on particular EDA tools. Hence, their presence, naming and usage are typically unique to a tool and are used for such things as specifying outlines, part borders, keep-outs, and for various design rule checking purposes (DRC). As you will learn, Allegro has many such special layers.

9. **Board Outline**. Every PCB design requires a planar drawing of the board's perimeter or outline. These vary from simple squares and rectangles to complex shapes involving arcs.

At this point it might be helpful to have a simple picture to look at. Shown is the 4-layer cross-section for an actual PCB in Allegro (later, I'll show you how to use this editor). First, observe this is not a true scaled mechanical drawing, since the insulating layers, the three *dielectric* layers, are actually much thicker than any of the four copper layers, the yellow and red *conductors* and green *planes*, typically from 8 to 60 mils or so. Copper traces on the four conducting layers typically have heights of only about 1 to 4 mils, depending on copper area density.



		Surface	
1	TOP	Conductor	Copper
		Dielectric	Fr-4
2	LAYER_1	Plane	Copper
		Dielectric	Fr-4
3	LAYER_2	Conductor	Copper
		Dielectric	Fr-4
4	BOTTOM	Conductor	Copper
		Surface	

Second, note that only physical layers having etched copper are shown. Logical layers that result in physical features, like solder masks and silk-screens, must be visualized between the top and bottom etch layers and the surface air above and below each respectively. This sideways cross-sectional view is the one you want to keep in mind while designing PCB's. Layout tools, like Allegro, always show the PCB as if we were geometrically looking down through the layered cross-section from the top along the Z-azis. Note the standard geometric orientation is based on the conventional right-handed Cartesion 3-axis coordinate system with all layers lying in the X-Y plane and the cross-section orthogonal to the Z-axis. All industrial references use this orientation.

Notice that Allegro allows us to specify what type of material is to be used for each layer. This example shows the routing and power planes with copper, and the dielectric layers as "FR-4" (generic glass-expoxy material). Not shown is the rest of the spreadsheet where we can specify thicknesses and associated electrical properties. These latter properties are primarily useful when we need to simulate electrical performance associated with particular designs. We won't be concerned with any of these in this laboratory, but you should know that Allegro has a powerful set of simulation tools meant for high frequency analog and high-speed digital PCB layouts.

I will discuss some of these features, but only those actually relevant to our immediate work, with more depth as we encounter them in the context of actually using Allegro.

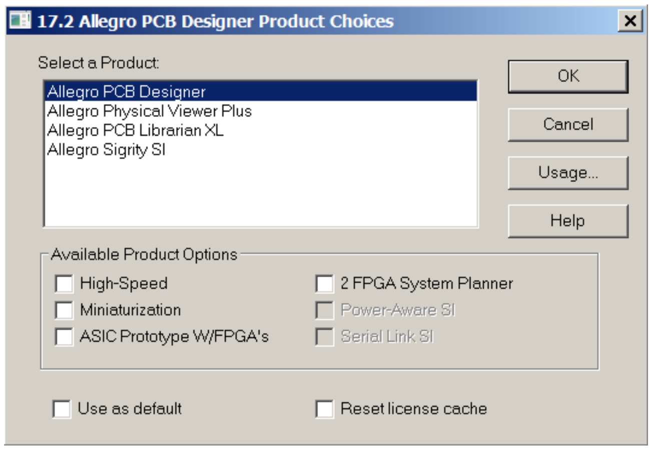
3. INTRODUCTION TO ALLEGRO PCB DESIGN

In this laboratory, physical PCB layouts are designed and managed using *Allegro PCB Designer*. Cadence markets this tool in various "tiers", each with different capabilities, but all having the same basic user-interface (UI); in fact, one can't typically tell which flavor is currently being used except either by looking at the top-most line of the program, or by knowing which specific icons and associated menu features are included or missing since the apparent differences are often very minor (but… the capabilities are *not* minor). We will be using the Designer version throughout. From this tool the entire backend set of basic, and some very advanced operations, are available. For example, this version has many sophisticated simulation and auto-routing features that we won't be using , but I will point some of them out as we go along so you won't be confused by their presence. These features are unavoidably complex and complicated and necessarily overlap with the fundamental features treated in this lab. Rather than ignore them as if they didn't exist, and you can certainly do sophisticated manual PCB layout from this perspective, I believe it is more useful to discuss them in the contexts they naturally occur so you can develop an understanding of their relative importance to what you actually want to do. For example, I've already mentioned we won't be using the autorouter, but while manually routing traces by hand, Allegro will in fact use some autorouting features by default; you need to know what these are, why they're there and how to use them effectively, even while only doing manual routing. I also believe this will serve you better in the future when you will most likely want to make use of them, especially in complex digital designs.

Sequentially then, this backend phase begins with a compiled netlist from Capture and uses it with Allegro to create a finished PCB board to be contained in a single board file (*.brd). From this finished design, we continue to use Allegro to proceed to the final steps of generating output "artwork" and drill files necessary for a board house to actually manufacture a bare PCB. Along the way, generally speaking, we might also use some of the advanced layout features to simulate, and thus predict, time and frequency domain consequences of particular layout topologies. Again, as already noted, we won't use these features, but you should know they exist, especially if you ever get involved with high performance, high frequency PCB implementations typically associated with RF and high-speed digital designs.

3.1 SETTING UP THE USER INTERFACE

Launch **Allegro PCB Design** from the main Windows menu where it may be labeled *Allegro Pcb Design* or *Orcad PCB Editor*. If the program boots without asking for a choice of versions (as it should), be sure it states *Allegro PCB Design L (Legacy)* across the very top. If it doesn't, change to the proper editor by menu sequence **File → Change Editor.** This will bring up the Product Choices dialog as shown. Note this may also appear automatically, unless a previous user has checked the "Use As Default" checkbox so this selection is then bypassed (Note the product choices will be for 16.6, not 16.2 as shown, but are otherwise identical).


17.2 Allegro PCB Designer Product Choices

Select a Product:

Allegro PCB Designer
Allegro Physical Viewer Plus
Allegro PCB Librarian XL
Allegro Sigrity SI

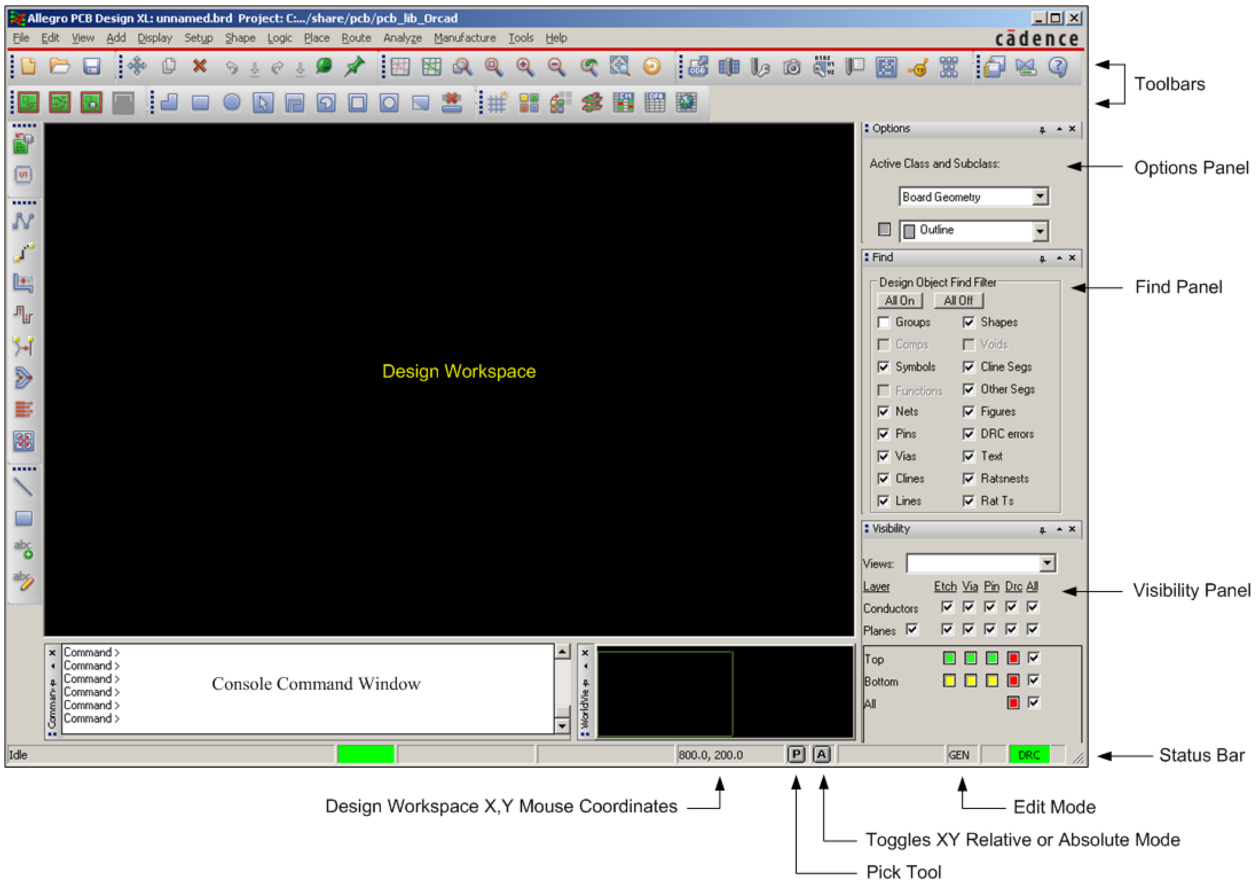
Available Product Options

- [ ] High-Speed
- [ ] Miniaturization
- [ ] ASIC Prototype W/FPGA's
- [ ] 2 FPGA System Planner
- [ ] Power-Aware SI
- [ ] Serial Link SI

- [ ] Use as default
- [ ] Reset license cache

Select "… PCB Designer", set the default checkbox so it will boot automatically next time and press Ok. The program should boot and look something like that shown on the next page.

The horizontal (those across the top) and vertical (those along the left edge) toolbars contain many icons grouped by function[2]. They are easily understood when classified according to function group. Their visibility and position can be defined as needed. Some of the icons don't have menu equivalents and so are only accessible by mouse selection. On the next page I show you which default toolbars should be visible and which shouldn't. Minimizing their presence makes more space available for the Design Workspace.



Note: the screenshot above is from PSB 16.2. PSB 17.2 is otherwise identical with some minor changes in where things are located on the status bar and the Options, Find and Visibility panels.

The **Options**, **Find** and **Visibility** panels are central to using this tool effectively. Configure them to be one atop the the other and dockable as shown. Their content changes depending on the working context in the Design Workspace; this is especially true with the Options and Find panels. *You should always remember that Allegro is a highly context-driven program*, so you must develop the habit of mentally noting what mode or combination of modes the tool is currently working in. Typically, there is no explicit indication whatsoever, except what appears on the Options panel, so failure to remember will lead to untold frustration at times when you want to do something but Allegro stubbornly won't let you.

The **Status Bar** keeps track of particular settings, like providing an instant XY indication of cursor coordinates anywhere within the Design Workspace. The most important of these settings is the master **Edit Mode** annunciator, currently showing Allegro to be in the General Edit Mode. Possible modes we will be using are: *General Edit*, *Etch edit* and *Placement edit*.

Some commands are more quickly entered from the **Console Command** window than by the usual menu selection sequence. Further, some commands also have no menu or icon equivalents and can only be entered by typing them in (there are also many commands found in the help resources that simply don't work). I suspect the genesis of this interface goes back historically to earlier versions when the interface was command-line driven. Today, just about

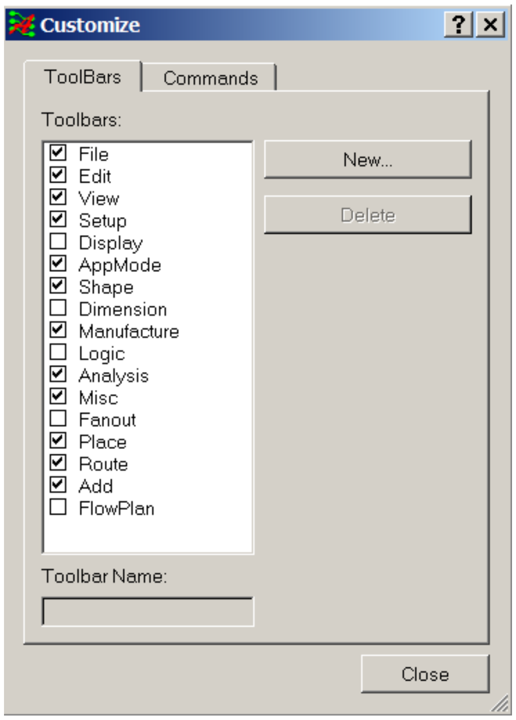
[2] See *Allegro PCB Editor Tutorial*, pg. 64 for a partial summary of these by named blocks. Accessing this document is described in Appendix-A, *Finding Help Resources*.

everything can be done using the mouse and GUI, but I have found that several significant action sequences can only be done using console commands; I will point these out when we get to them. Also, by typing *helpcmd* a listbox appears in which all actual possible commands are alphabetically listed. You can select any of these to execute or obtain help.

The **Worldview Window** isn't very useful for small single-board projects, but otherwise gives a bird's eye view of where in the overall total area the Design Window is presently working, especially when highly magnified. It can be used to pan around at will by drawing rectangles using the mouse (click-and-drag).

Finally, visibility and position of these several panels and windows are controllable from the three icons present on the top line of each one; how these work can be quickly understood by judicious experimentation. I usually turn off the Worldview and tack the Console (using the "thumbtack" icon) making it a popup when needed; this increases the Design workspace area. Dynamically set these features to best suit yourself as you work in different contexts.

Setting the Toolbars – As I mentioned above, only displaying needed toolbar groups will save space, unclutter the desktop and make more area available for the real tasks at hand to be done in the Design window. Do this from menu sequence **View ⟶ Customize Toolbar…** This will bring up the Customize dialog shown.


Customize

ToolBars | Commands

Toolbars:

- ✓ File
- ✓ Edit
- ✓ View
- ✓ Setup
- ☐ Display
- ✓ AppMode
- ✓ Shape
- ☐ Dimension
- ✓ Manufacture
- ☐ Logic
- ✓ Analysis
- ✓ Misc
- ☐ Fanout
- ✓ Place
- ✓ Route
- ✓ Add
- ☐ FlowPlan

New... | Delete

Toolbar Name:

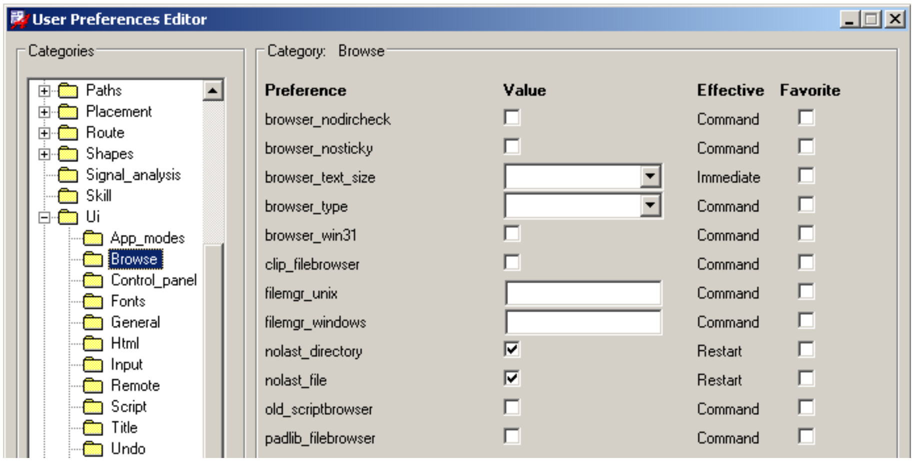
Close

Since I forgot to check the Display toolbar group, do so now and observe it immediately appears on the desktop. Check and uncheck each of the others to quickly identify icon groups and close the dialog.

User Preference Editor

We'll need to configure a number of default user paths so Allegro will properly recognize your folder tree, and to specify several user interface preferences. This is all be done within the *User Preferences Editor* dialog shown below. Access the editor by menu sequence **Setup → User Preferences…**

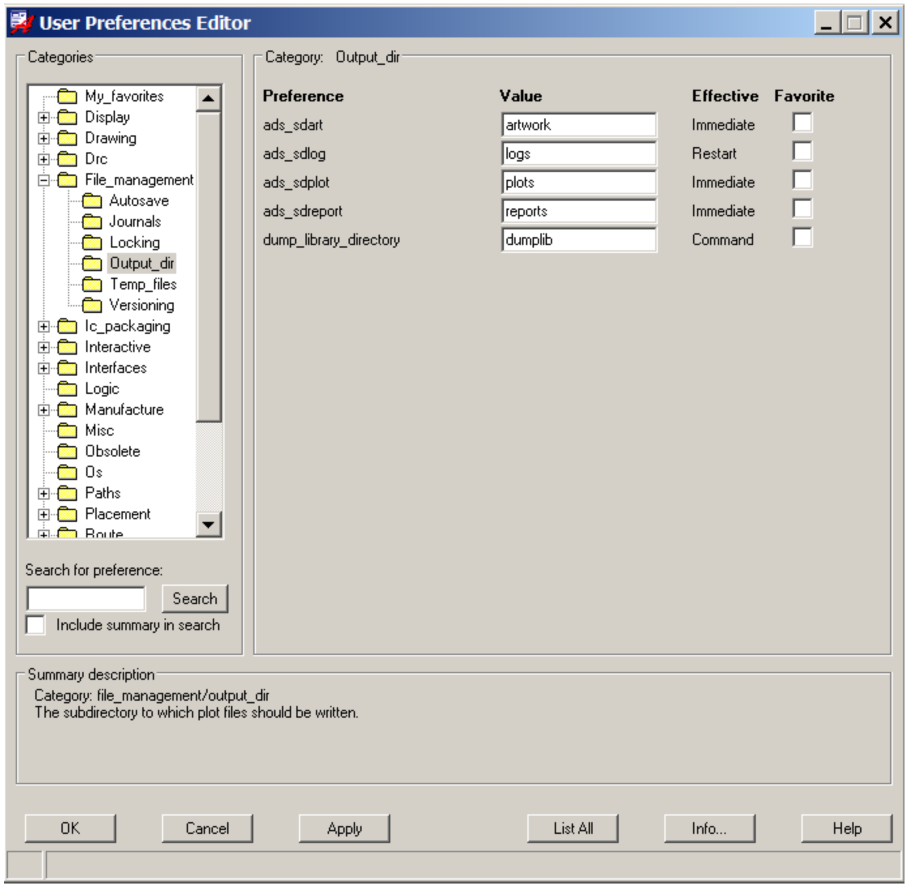
1. First, select virtual folder *Ui* / *Browse* and check the the *nolast_directory* and *nolast_file* as shown. This will prevent Allegro from trying to access and reload the last used directory and board file. Since many students will be using Allegro having different paths to their own files, we need to configure Allegro to *not* remember anyone's last used directory and file.



Category: Browse

Preference	Value	Effective	Favorite
browser_nodircheck	☐	Command	☐
browser_nosticky	☐	Command	☐
browser_text_size		Immediate	☐
browser_type		Command	☐
browser_win31	☐	Command	☐
clip_filebrowser	☐	Command	☐
filemgr_unix		Command	☐
filemgr_windows		Command	☐
nolast_directory	☑	Restart	☐
nolast_file	☑	Restart	☐
old_scriptbrowser	☐	Command	☐
padlib_filebrowser	☐	Command	☐

2. Second, select *File_management / Output_dir* and enter the folder names as shown for each of the output directories. This directs Allegro to write related file types to these folders, relative to the working pcb directory, which should be …\Tutorial\Handset\pcb as shown back on page 3.



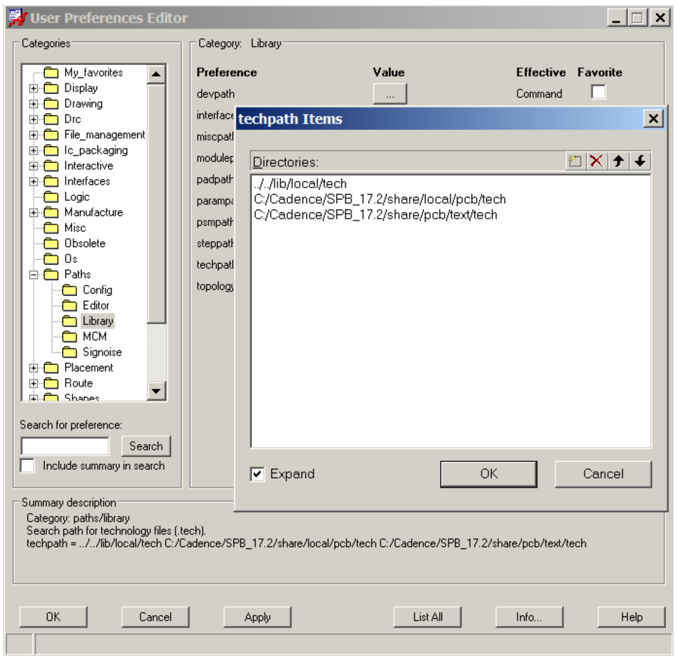
Category: Output_dir

Preference	Value	Effective	Favorite
ads_sdart	artwork	Immediate	☐
ads_sdlog	logs	Restart	☐
ads_sdplot	plots	Immediate	☐
ads_sdreport	reports	Immediate	☐
dump_library_directory	dumplib	Command	☐

Summary description
Category: file_management/output_dir
The subdirectory to which plot files should be written.

Look over the *Temp_files* and *Versioning* options. We'll leave them as they are.

3. Third, default search paths are set in the *Paths* subcategory. We're interested in expanding some of the paths found in the *Library* category shown below. Each of these can be edited by pressing their associated *Value* button: ...



```
../../lib/local/tech
C:/Cadence/SPB_17.2/share/local/pcb/tech
C:/Cadence/SPB_17.2/share/pcb/text/tech
```

☑ Expand

Summary description
Category: paths/library
Search path for technology files (.tech).
techpath = ../../lib/local/tech C:/Cadence/SPB_17.2/share/local/pcb/tech C:/Cadence/SPB_17.2/share/pcb/text/tech

For example, I did this for the *techpath* preference, checked the *Expand* checkbox, and then removed the existing relative paths at the top and added the new relative path line literally as shown: **../../lib/local/tech**

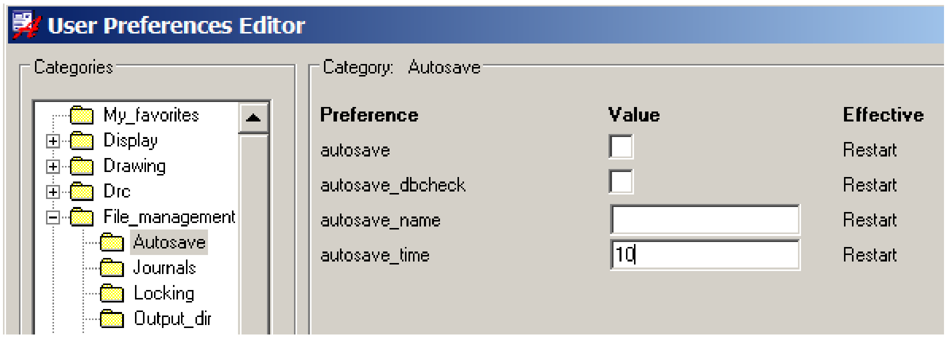
In addition to the Cadence default paths shown, this line adds a relative path to your personal technology files.

Below I've listed all the paths that need to be similarly edited (including the one I just did above); add the fields as shown.

techpath	**../../lib/local/tech**
modulepath	**../../lib/modules**
padpath	**../../lib/padstacks** **../../lib/symbols**
parampath	**../../lib/local/parameter**
psmpath	**../../lib/symbols**

TIP: For these relative paths to work correctly and map to existing folders in your project folder tree, the working directory must be **../Tutorial/Handset/pcb**. Be careful when using the folder or directory browser dialogs in Allegro, since many give you the option of "Changing Directories" at the bottom of the form. Checking this box will immediately change the working directory (after pressing OK of course). See an examples on page 20.

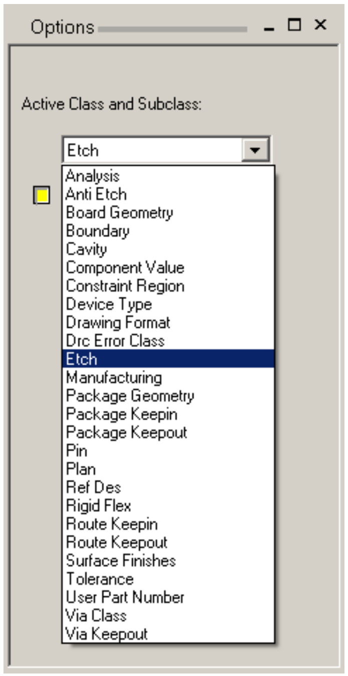
Configuring Autsoave – The last category to configure is to enable *autosave* and set Allegro to make a backup of the current board file every 10 minutes. Set the *File_management* / *Autosave* category fields as shown.



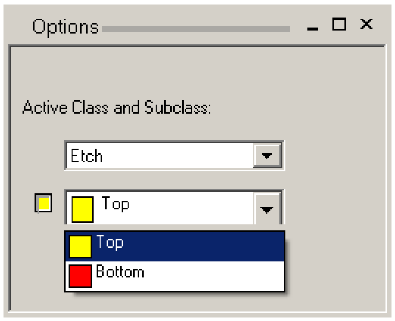
Preference	Value	Effective
autosave	☐	Restart
autosave_dbcheck	☐	Restart
autosave_name		Restart
autosave_time	10	Restart

3.2 Allegro Classes & Subclasses

Allegro's architecture employs a peculiar set of names for the traditional generic layers described earlier, reminiscent more of a computer scientist's view of the world rather than an engineer's. You will need to fully understand this to competently work with this tool: Every visible drawing element, like board outlines, copper layers, silk-screens etc. are represented through the taxonomy of various *classes* and *subclasses.* For example, physical layers having actual copper planes or routed traces are classified as subclasses under the parent *ETCH Class*.



I've shown in the pull-down menu left, within the Options panel, all of Allegro's defined master Classes and have particularly selected the ETCH class to be the "active class", namely the one I want to work with now, in the Design window. After making this class active, the pull-down list immediately below always contains any subordinate subclasses. Since the Etch class defines actual physical copper layers in a board's cross-section, for a simple 2-layer stackup there will only be two of them as shown below for the *Top* and *Bottom*, where I have selected the Top to be active.

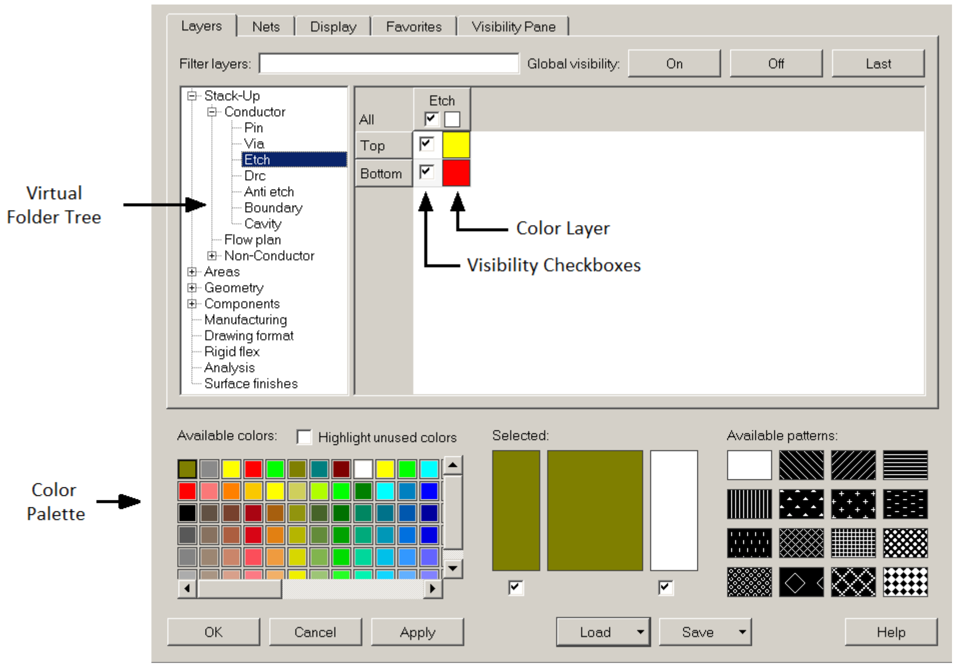


The yellow square immediately to the left of the selected subclass is used to toggle that layer's visibility by clicking it with the mouse. You will quickly discover that many of the member subclasses are special logical layers that don't directly describe any of the classic logical or physical layers in a generic PCB's stackup (a good example is the Drawing Format class), but so long as you remember they do refer to some *visible drawing element* that appears for some, often obscure, purpose in the Design window, then you won't get confused.

Color Dialog

You probably noticed the two colors in your Option's panel are different than mine, most likely Cadence's green and yellow. I prefer different colors, and you probably will too. Color changes to any sub-class can be made here, but this isn't very convenient when we want to change many at once. There is another dialog window you need to know about that is closely related to the Active Class and Subclass views in the Options panel: the *Color Dialog*; this is where colors and visibility, among several other important things, can be defined in a large spread-sheet format. Access this by menu sequence **Display → Color/Visibility** or use the faster toolbar icon. This will bring up the master *Color Dialog*. Here you can change and set colors and control the visibility of entire classes and sub-classes, whereas in the Option's panel this can be done only on singly selected sub-classes. Beginners find the relationship between the Color Dialog and the Options panel understandably confusing, so I'll spend a few moments on it. If you don't understand all of this now, that's ok. After gaining some real experience with Allegro, refer back to this part of the lab for clarifying help. Checking the "Layers" checkbox at the top causes a virtual directory tree to appear that organizes classes differently than in the Options panel. I have deliberately selected the Etch class so you can see the differences between these two views of the same class (screen shot is on the next page).

With the exception of the first two virtual folders, *My Favorites* and *Display*, all the rest are either single classes or represent logically related groupings of classes. Look carefully at the list in the example shown. Notice only three contain sub-folders: *Stack-Up*, *Areas* and *Components*. Each of these group logically related classes but are not themselves classes, while all of those having no sub-folders are actual classes. Somewhere within this set of folders all of the classes seen in the pull-down list in the Option's panel above can be found.



Visibility is controlled by the checkboxes in the edit window. Their use is obvious after a little experimentation. The *Global Visibility* buttons in the upper right apply, as the title suggests, to all sub-classes found in the virtual folder tree; and this means *everything*. So be careful using this since the operation of selecting *On* or *Off* followed by *Apply* is irreversible!

I should also warn you that working with the Color Palette has some annoying bugs. Colors to sub-classes are mapped by relative row/column position within the palette, so we're supposed to be able to export and import palettes; unfortunately it doesn't work properly. Later on, I will discuss how to use this dialog with the alternative choice, the "Nets" option after we have some real nets to work with in the design window. My purpose in showing you the Color Dialog window here is to emphasize the very close relationship between it and the Active Class and Subclass views of the same objects in the Options panel. The Color Dialog is used to set things up or to do things that can't otherwise be done in the Options panel.

All of the classical layers I mentioned earlier can of course be found as one or more subclasses under appropriate parent classes. These are mapped to class / subclasses as follows:

Top-side Copper:	The following three combined represent this layer. Etch / Top (for traces) Pin / Top (for padstacks) Via Class / Top (for vias).
Top-side solder-mask:	Pin / Soldermask_Top
Top-side paste-mask:	Pin / Pastemask_Top
Top-side silk-screen:	Some combination, one or more of the following: Board Geometry / Silkscreen_Top Component Value / Silkscreen_Top Device Type / Silkscreen_Top Package Geometry / Silkscreen_Top Ref Des / Silkscreen_Top Tolerance /Silkscreen_Top User Part Number / Silkscreen_Top

The above four layers are all repeated for the bottom-side.

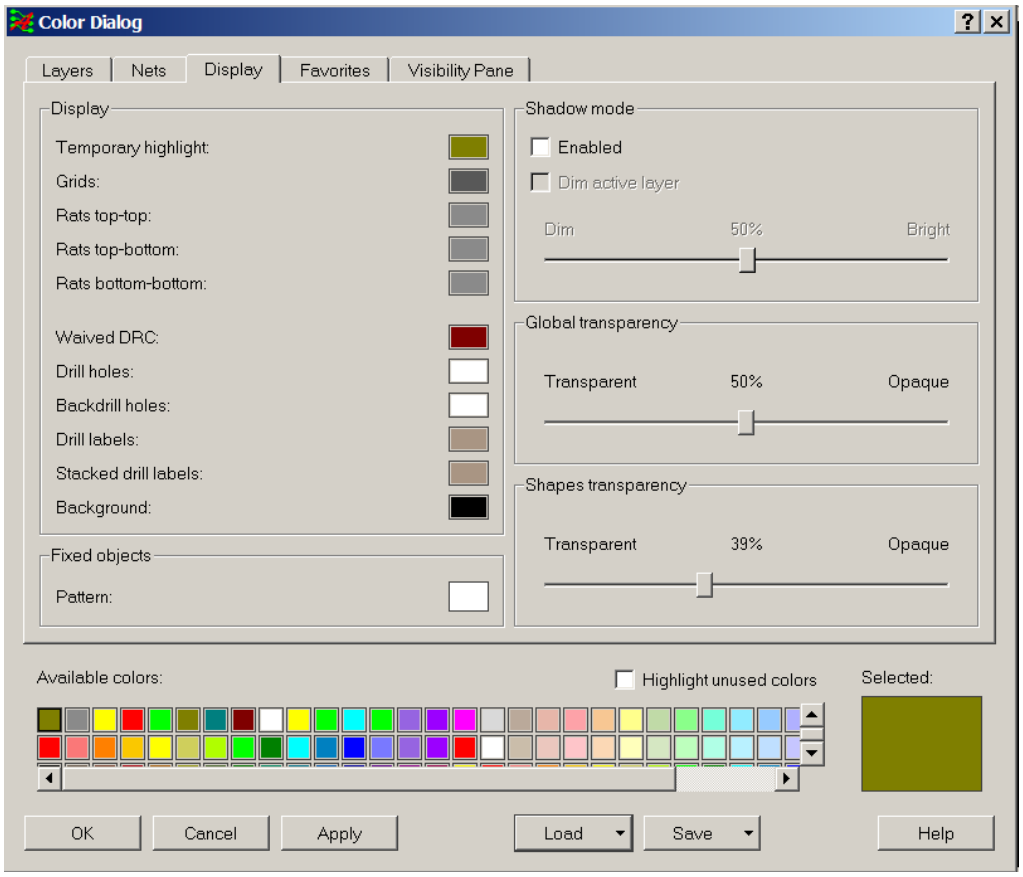
Any inner layers, if they exist, will be mapped the same way as Top-side Copper. For example

Inner-1:	Etch / Inner-1 Pin / Inner-1 Via Class / Inner-1
Drill targets.:	Manufacturing / Ncdrill_Figure
Board Outline.:	Board Geometry / Outline.

What should be evident is the great versatility of this classification scheme. The final silk-screen, for example, can be formed from seven possible subclasses; generally, we only combine several of these as you will see at the end of the lab.

Working with the Display

Allegro has a sophisticated set of features to control the intensity and transparency of the visual display in the Design Workspace. Only systems having video cards capable of OpenGL can transparency be set to less than 100%, to "see through" lower layers in the context-sensitive z-axis ordering (whichever layer is currently active is always the highest). Since our lab computers have OpenGL, we'll be using these features. Understanding how to effectively use them can be only be appreciated experimentally, so at this point, I just want to survey the controls and initially set them for the work we'll immediately be doing. OpenGL settings are found on the Color Dialog in the "Display" virtual folder. Access this now and set the *Global Transparency* to 100% (solid). This causes the footprints you'll soon be importing to appear as bright as possible. Notice the other options in this class: *Shadow mode*, *Shapes transparency* and *Display* settings. Later, after having added some routed traces and copper areas, I'll discuss how they're used; for now, set Shadow mode off and Shapes transparency for 39%. The Display colors should be like those seen. If not, change them using the color pallete (cut-off in this screen shot at the bottom).



Display

Temporary highlight:	🟩
Grids:	⬛
Rats top-top:	⬜
Rats top-bottom:	⬜
Rats bottom-bottom:	⬜
Waived DRC:	🟥
Drill holes:	⬜
Backdrill holes:	⬜
Drill labels:	🟫
Stacked drill labels:	🟫
Background:	⬛

Fixed objects

Pattern: ⬜

Shadow mode

☐ Enabled

☐ Dim active layer

Dim 50% Bright

Global transparency

Transparent 50% Opaque

Shapes transparency

Transparent 39% Opaque

☐ Highlight unused colors

3.3 THREE PRECONFIGURATION FILES

At this point you might correctly suspect there are many tedious preconfiguration steps necessary to setting up Allegro before we even begin the task of importing the netlist. Fortunately, we don't have to do this for every new board, but can import many different files that make this task easy. Among these the three most important to understand and use effectively are the following: *Template*, *Parameter* and *Tech* files.

Templates

These are customized but otherwise completely empty PCB board files (*.brd) that serve as good advanced starting points. Specifically, these files expedite the drudge related to otherwise having to manually define the following:

- **Drawing area size**. Sometimes called *drawing extents*, this is the maximum sandbox area within which *everything* must be enclosed, including any documentation layes, as well as the PCB itself. Allegro's default extents are 34 (width) by 22 (height) inches. For small boards we typically use this default size.
- An optional **board outline** defining the actual PCB's physical perimeter. If it's present it can easily be modified; if not, it must be added since every layout requires one *before* the netlist can be imported.
- **Spacing constraints**:
    - Minimum line (copper trace) width.
    - Minimum line to line spacing.
    - Minimum line to pad spacing.
    - Minimum pad to pad spacing.

  These serve two purposes. First, they specify how close etched objects can be before DRC errors will be flagged. Note the minimum trace width constraint is determined by the design's complexity level (discussed below). Second, they are automatically enforced while we manually route the new board by hand. These represent only a subset of all spacing constraints; one example of others you will need to consider is how close footprints can be to one another, but these form a basic set.
- **Package and route keepins**. *Keepins* delimit areas where packages (footprints) are permitted to be placed and where traces (routes) can occur. These parameters are mostly for automatic moving, placement and routing routines - features we won't be considering or using in this laboratory.
- **Grid definitions**. Grids are necessary for placing footprints, routing nets and placing vias. Allegro defines two basic grids: non-etch and etch. How these work can only really be understood and appreciated in the context of actually working with them.
- Layout **cross-section** (stackup).
- Customized data, such as default **parameter settings**, default **subclasses**, and **color assignments**.

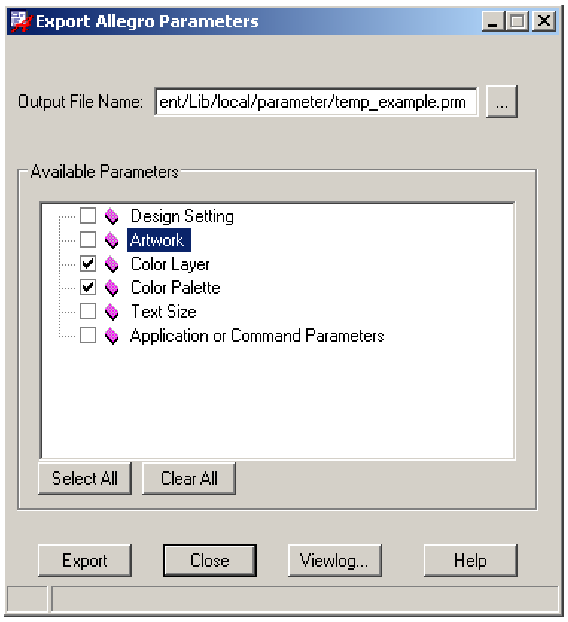
Although a board outline may be present in a template, they are still considered "empty" so long as there are no footprints, traces, pins or vias; this means they can have nothing drawn or placed in any of the subclasses subordinate to the ETCH, PIN or VIA classes.

Design Complexity - Every PCB has an intended *design complexity*, classified as Level A, B or C, and are meant to be guides corresponding to the relative ease and related cost of manufacture[3]. There are many physical constraints that must be set for each of these levels that are tedious to set up by hand for every board, so defining them in template files is very convenient. Level A has the least stringent parameters, allowing route spacing no closer than 12 mils (0.012 inch) – which is pretty wide for clearances between traces, pads etc. Level B, considered the "standard" reference, reduces this to 8 mils, and C to 6 mils. A typical board house can produce boards for level A or B at the same cost; beyond that it goes up quickly, and some board houses won't do level C board runs at all. We will be designing our board for level B; this is noted in the template filenames above.

[3] See *The Institute for Interconnecting and Packaging Electronic Circuits*: IPC-2221, Generic Standard on Printed Board Design, sect 1.6.3; and, also IPC-2222 dealing with Rigid Organic Printed Boards. Both of these are in the Appnotes folder.

Parameter Files (*.prm)

These contain information related mostly to customizable “look-and-feel” aspects of a PCB design and various behavior options. This includes global design settings such as dynamic fill; grids; artwork format; XHatch styles, line widths, spacing, angles, text size etc. Behavior examples include things like automatic part placement; global dynamic fill, glossing etc. Most of these won’t mean much to you until after you’ve had a chance to go through a complete PCB design, so don’t worry about trying to understand them with much depth or completeness now. That will naturally follow actually creating and using them. These files are found by default in the shared installation folder: **…SPB_16.6\share\local\pcb\parameter,** and in your folder tree: **…Tutorial\lib\local\parameter.**


Export Allegro Parameters

Output File Name: ent/Lib/local/parameter/temp_example.prm

Available Parameters
- [ ] Design Setting
- [ ] Artwork
- [✓] Color Layer
- [✓] Color Palette
- [ ] Text Size
- [ ] Application or Command Parameters

Parameter files are created by exporting them from existing board designs (*.brd). Typically, this is done when we want to re-use a subset of existing features already in another design by following menu sequence **File⟶Export ⟶ Parameters…** This brings up the *Export Allegro Parameters* dialog shown.

In this dummy example, I’ve arbitrarily checked two of the possible seven available parameter categories (one is not shown – *Color Net*) with the intention of exporting them to output file *temp_example.prm* to be written in my project’s **…lib/local/parameter** folder. Follow this example should you ever need or want to do something similar (and you will).

For future reference, I’m including below a brief summary of what parameters the six most useful groups encapsulate (selecting *Help* accesses Allegro’s reference I based this on):

1. **Design Setting.** Global values and grid settings.

2. **Artwork.** In Allego, this term refers to the output files necessary to actually manufacture a bare PCB by an external boardhouse. Traditionally, industry calls these Gerber files, and they are created during the *post-processing* phase at the very end. Since the original Gerber process involved photographic film steps, these are also sometimes referred to as “films”. The point to remember here is that if you want to save your manufacturing parameter settings then check this box.

3. **Color Layer.** Color parameters and color table.

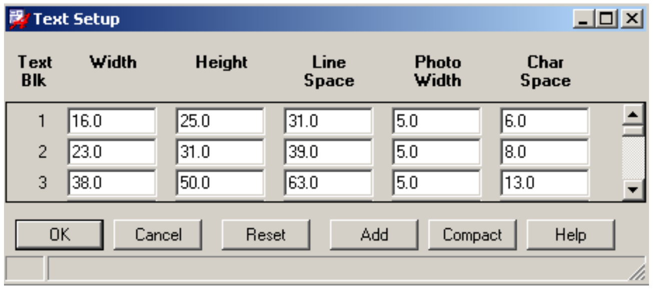
4. **Color Net.** This selection is not shown, but it would appear if a netlist had been loaded and specific colors were assigned to any of the associated nets. Cadence tells us this exports any net’s custom color and state (visibility). “When a file containing net color data is imported into any design, only the nets that exist in that design are read; the rest are ignored. Net color assignments are not overwritten, but rather incremented. To completely replace net color assignments, click *Clear All Nets* in the *Nets* section of the Color dialog box before importing a file containing net color data.” For example, this feature might be useful if commonly occurring nets, like Gnd and VCC should always be assigned certain colors and visibility.

5. **Color Palette.**

6. **Application or Command Parameters.** “All other supported parameters, including those for auto rename, auto assignment, auto silk-screen, global dynamic fill, autovoid, export logic, drafting, gloss line fattening, gloss dielectric generation, Options window tab settings, test prep, automatic placement, auto swap, thieving, backdrill, interactive flow planner (Allegro only), and Signoise analysis.”[4]

[4] Quoted from “define text” in Allegro PCB and Package Physical Layout Command Reference.

7. **Text Size.** Although seemingly insignificant, properly writing text on different layers, particularly documention layers like silk-screen can be a real pain. This option exports the entire current text table shown below. Note that each text block (*Text Blk)* is referenced by number having five associated properties per entry (I've shown only three of the 17 entries); how this is used will be discussed later. The ratio of width and height to photo width must be correct to give a readable pleasing appearance, as well be physically printable on a PCB. Custom entries can be added to this table if you like. Note this table is accessed by menu sequence **Setup⟶Design Parameters…** choose tab: **Text**, then check the *Setup Text Sizes* checkbox. The screen shot shows only the first three text block definitions.


Text Setup

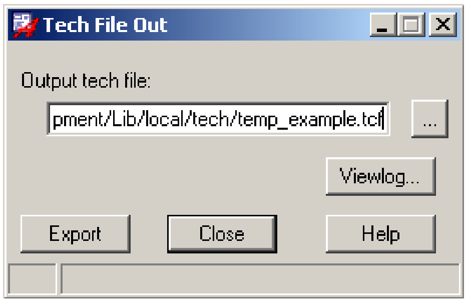
Text Blk	Width	Height	Line Space	Photo Width	Char Space
1	16.0	25.0	31.0	5.0	6.0
2	23.0	31.0	39.0	5.0	8.0
3	38.0	50.0	63.0	5.0	13.0

Technology Constraint Files (*.tcf)

These (also known as "tech" files) contain the following:

- *Drawing parameters*, such as units (*viz.* mils, metric or inches).
- *Layout cross-section.*
- *Design Rule Checks.*
- *Complete Design Constraints* for various physical and electrical specifications, including the spacing constraints defined in templates.
- *User properties.* These are custom properties defined by you.
- Found by default in the shared installation folder: **…SPB_16.6\share\local\pcb\tech** and in your folder tree **…Tutorial\lib\local\tech.0**

Tech files are created by exporting them from an existing, typically finished, board design (*.brd) through the same procedure noted above for parameter files, namely by following menu sequence **File⟶Export ⟶ Techfile…** This brings up the much simpler *Tech File Out* dialog shown.


Tech File Out

Output tech file:

pment/Lib/local/tech/temp_example.tcf

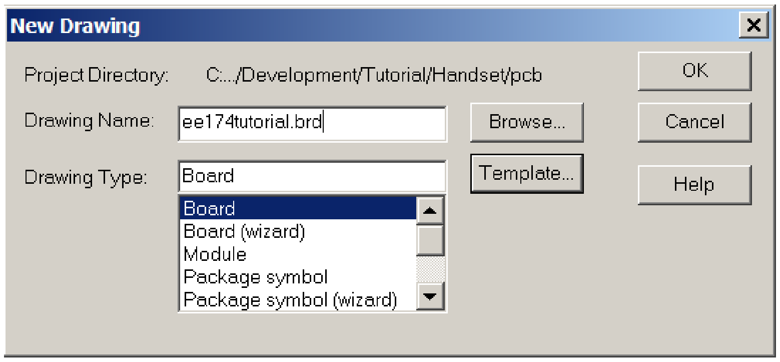
Again, in this dummy example, I navigated to my project tree's tech folder to save it in: **…Lib/local/tech**.

It should be obvious these three configuration files can be mixed and matched in versatile ways, depending on what we want to accomplish. Experienced users may employ board templates to expeditiously create basic design frameworks that can serve as foundations for more complex designs capable of being further customized by *parameter* and *technology files;* these would have been previously created by you specifically for this purpose. For example, we could begin with a level B design and change it to level C without changing anything else by simply importing a suitable technology file. I should also add that Cadence help resources are particularly vague about the precise contents of these three file types, but it appears that a properly defined template file *should* be a proper superset of everything that could also be defined in parameter and tech files. This follows logically from the fact that parameter and tech files are exported from board files that could just as well be made into templates by removing all the etch layer content.

3.4 CREATING A NEW BOARD

Two ways exist to begin a new board layout, manually or with the wizard. We will create and configure our new 2-layer board file manually by first creating an empty design and then proceed to configure it by importing parameter and tech files *2_layer_rev1_.prm* and *2_layer_B_rev1_.tcf* ; both of these are in your project's lib folder under their appropriate sub-folders.

Do this with menu sequence **File** ⟶ **New….** This will bring up the *New Drawing* dialog as shown below. Choose the first option: *Board.* Then browse to your **…Tutorial\Handset\pcb** folder and enter the new drawing's filename as *ee174tutorial.brd.* Be sure to check the *Change Directory* option at the bottom of the navigation window to insure the Project Directory gets changed to the pcb folder. Press OK and then immediately save this new board file to disk: **File** ⟶ **Save**.

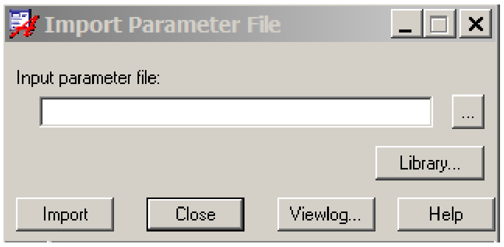


Project Directory: C:.../Development/Tutorial/Handset/pcb
Drawing Name: ee174tutorial.brd
Drawing Type: Board

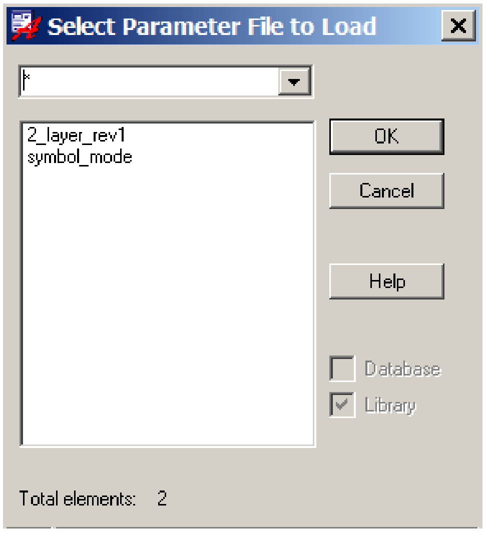
Note that the *Template* option can be used to begin new board files from an existing template file as a rapid time-saving feature. Consider using this in the future.

At this point we have the equivalent of an empty *.brd file with everything configured to Allegro's default values and settings. Proceed to configure this as a 2-layer design complexity level-B (8mil spacing constraints) PCB using a parameter file.

IMPORTING THE PARAMETER FILE



Follow menu sequence **File** → **Import** → **Parameters…** This will bring up the *Input Parameter File* dialog shown. At this point parameter files can be accessed in two different ways, by manually navigating to a folder containing them or by pressing the *Library…*button. Use of the Library option requires that *parampath* be properly edited. Since this was defined earlier select press the *Library…* button. This will go directly to you local **../../lib/local/parameter** folder and display the two parameter files found there.



From the File to Load dialog select the **2_layer_rev1** file. This will bring you back to the *Import Parameter File* dialog above. Press *Import* to load the new parameters.

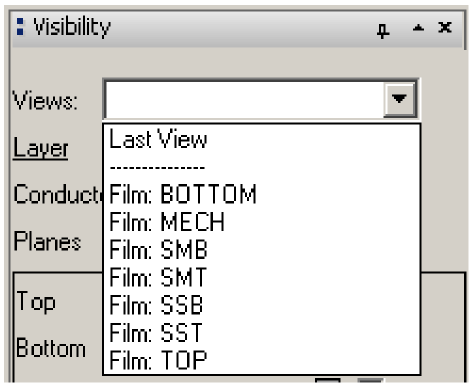
Note: if the manual browser button is used to navigate to your parameter folder instead, you would then select **2_layer_rev1.prm**. If you do this in the futre, be sure *not to check* the *Change Directory* checkbox to avoid changing the working directory.

IMPORTING THE TECHNOLOGY FILE

Loading the tech file follows a similar sequence of steps as for the parameter file. Follow **File → Import → Techfile…** Press the *Library* browser button and open the *2_layer_B_rev1* tech file. Like the parameter file, this will bring you back to the *Tech file In* dialog. Press the *Import* button to load the technology settings.

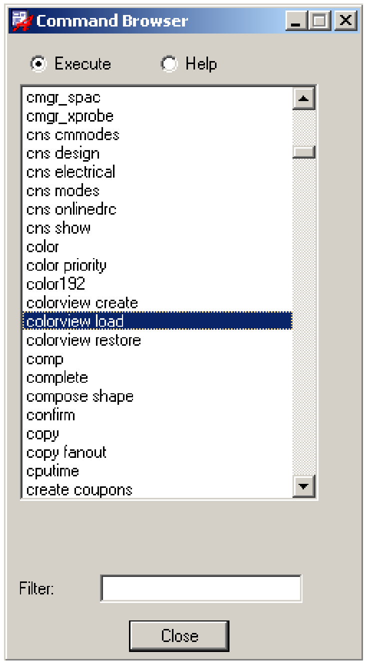
Adding Custom Colorviews

Colorviews refer to particular wanted combinations of visible layers (sub-classes) and are functionally divided into *Film* views and *File* views. Film views are used to create final *artwork* files meant to be sent to a “boardhouse” for PCB manufacturing. File views are used by us to selectively display wanted layers during the layout phase. A set of Film views necessary view and produce the classic layers for a 2-layer PCB was imported earlier in the parameter file. Although you don’t yet know how to use them, these named views can be turned on and off by selecting them from the collective pull-down *Views* list on the Visibility panel. The screen shot following lists seven Film colorviews defined in the imported parameter file we will need for our new 2-layer PCB:



BOTTOM:	Classic bottom copper (etch) layer.
Mech:	A special boardhouse documentation layer.
SMB:	Solder-mask bottom.
SMT:	Solder-mask top.
SSB:	Silk-screen bottom.
SST:	Silk-screen top.
TOP:	Top copper (etch) layer.

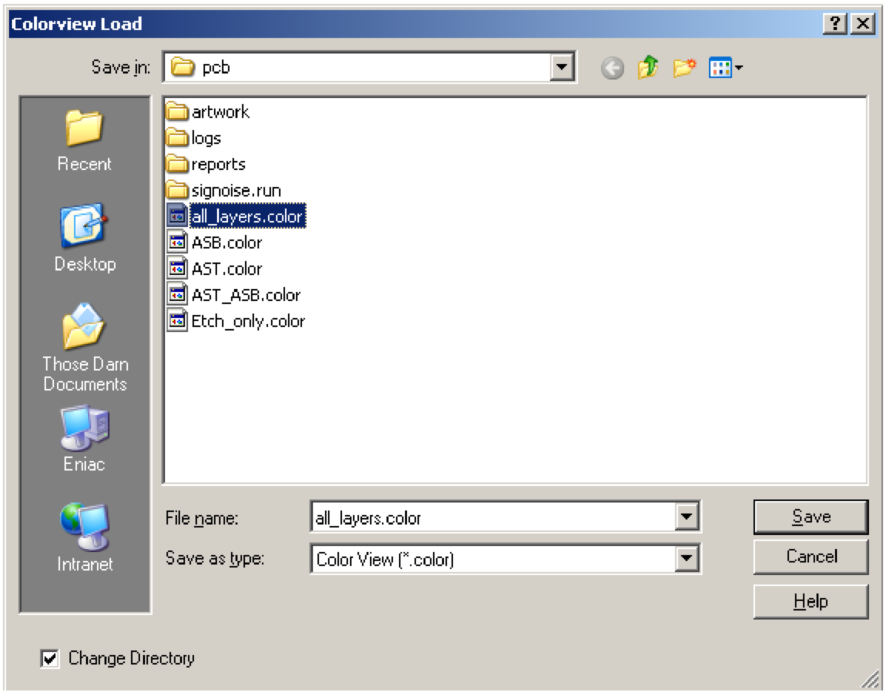
Besides affording us a way to make any of them visible, each will also be used in the last “manufacturing” step to produce Gerber files a board house needs to actually make the final physical 2-layer PCB. As noted above, Colorviews specifically for this purpose carry the prefix “Film” as shown.



Several additional “non Film” or File colorviews that cannot be saved in parameter files are also needed. I call these “working” views, since we need quick ways to make all sub-classes visible, and to make several different subsets visible while we’re actually placing parts and routing. These additional views are configured by loading custom *color* (*.color) files. This can only be done via the command console with the *colorview load* command. We’ll load these now.

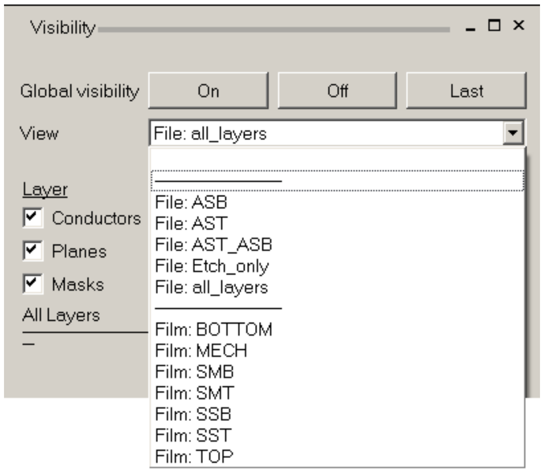
You could, of course, just type this command directly, but since you’ll probably forget it I’ll show you another way that uses the *Helpcmd*. In the Command Console access the command list by entering *Helpcmd*. This will bring up the **Command Brower** shown. Scroll down the list until you see the six or so color commands; select and execute the *colorview load* command by simply selecting it. Navigate to your **…/Handset/pcb** folder and load *one* of the *.color files found there; oddly enough Allegro responds by loading all of them (this may be a bug, but it’s really convenient). I notice in 17.2 that when the parameter file is loaded the colorviews are automatically loaded as well.

I have included a screen shot of the *Colorview Load* browser dialog showing the files that must be loaded. Due to a bug in Allegro, we must be in the default working directory for this command to recognize any *.color file. Hence, I have placed these files in what will be our working directory, so be sure that *Change Directory* is checked. If it’s not already set, this will specify our working directory now and allow the color files to load properly.



Successfully loading these color views will result in several new "File:" entries being added near the top of the *Views* list. As you work along, being able to turn these various layer combinations on and off is very convenient.

Colorviews are easy to work with. If you have a particular combination of visible layers that you find yourself using frequently, and setting them up in the Color Dialog is cumbersome, just save them as a named colorview using the complementary command: *Colorview create*. Because of the bug I mentioned earlier, be sure to save them in the same directory where the current working board file (*.brd) is found.



Defintions of what they do are:

ASB	Assembly bottom.
AST	Assembly top.
AST_ASB	Top and bottom assembly layers
Etch_only	No assembly layers, only etch.
all_layers	self-explanatory,

The AST & ASB views are basically for routing. They cause part outlines and reference designators to appear, making it much easier to identify nodes and related nets. Note that I also enabled the visibility of layers other than those included in the File views, but you'll have to use them to see why I've done this. The fileview at the very top is the last view open before accessing this pull-down list.

4. USING ALLEGRO PCB EDITOR

Master Mode Selection

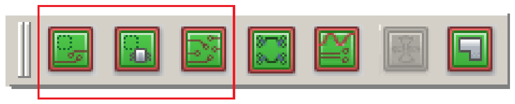
Before going on, we need to say a few things about Allegro's three master modes: *General*, *Etch Edit* and *Placement Edit*.

- **General Edit –** Enables us to perform generic tasks, like moving, copying or mirroring. (menu sequence: **Setup → Application Mode → General Edit)**.

- **Etch Edit -** Customizes the environment to specifically perform etch-editing tasks, such as manually routing traces, adding and sliding connections, delay tuning and smoothing cline (connect-lines; traces) or cline segment angles. (menu sequence: **Setup → Application Mode → Etch Edit).**

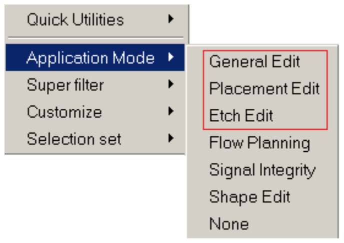
- **Placement Edit -** Customizes the environment to fine-tune placement and alignment of components and text. It also allows replication of circuits based on common connectivity and devices. A dockable placement window tab appears in the Control Panel conveniently providing immediate access to useful placement options. These include placing by group (that is, refdes, module instances, and so on) and filtering based upon property, room, part number, and net. (menu sequence **Setup → Application Mode → Placement Edit).**

Besides the menu sequences just noted, there are two much faster ways to set these modes. The first is by selecting them directly from the tool icon group shown:

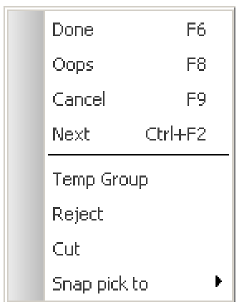


We are only interested in the first three shown within the red rectangle; from left to right respectively they are: General edit, Placement edit and Etch edit. Which one is active is also noted in the status bar where they can also be selected by simply clicking on the status bar label.

Beside the two ways just noted, another way is to right-click anywhere in the Design Workspace to access the context-sensitive menu shown. Note this only works if you aren't in some particular sub-context. In that case a different menu appears directly related to the sub-context, like change, move, add, delete etc.



Tip: If you *aren't* in a sub-context mode, right-clicking and seeing the above menu is a quick way to confirm where you are. If you *were* in a pending mode, *i.e.*one that you have not yet completed or exited from, you would typically see the following menu instead:

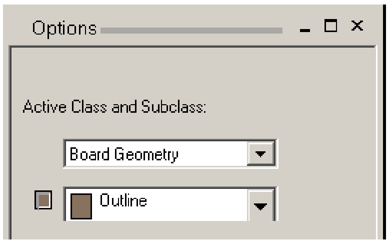


The top four options, *Done*, *Oops* etc., always appear, although some may be greyed out, while the bottom set changes depending on the actual sub-context; the one shown is assocated with deleting things. After completing an action, right-clicking and selecting *Done* is a very frequent sequence; the other, as you might suspect, is *Opps* (simply exits without doing anything). I like this bit of humor, since we seem to choose Opps as often as Done ☺ (note too the hotkeys…). Cancel means, "I know where I am, but I want leave", while Opps means, " I'm not sure where I am, but get me outa here!"

4.1 ADDING A BOARD OUTLINE

Before Allegro will allow us to import the netlist and begin designing the PCB, a board outline must be drawn on the **Board Geometry / Outline** layer, since footprints are place with respect to it.

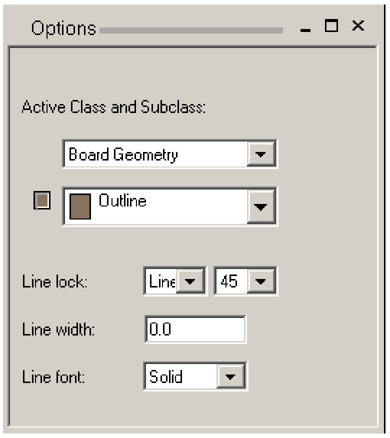
Create a board outline 25 mils thick on the Global Layer, 2.000 inches wide by 6.000 inches high on the Outline sub-class somewhere in the middle of the working area. Do this as follows:



Active Class and Subclass:
Board Geometry
Outline

Set General Edit mode. Set the Active Class and Subclass in the Options panel as shown, and be sure XY mode is Relative (the "R" annunciator should appear on the status bar at the bottom of the screen). Refer to the screen shot and discussion on page 6 if you need help with the status bar.

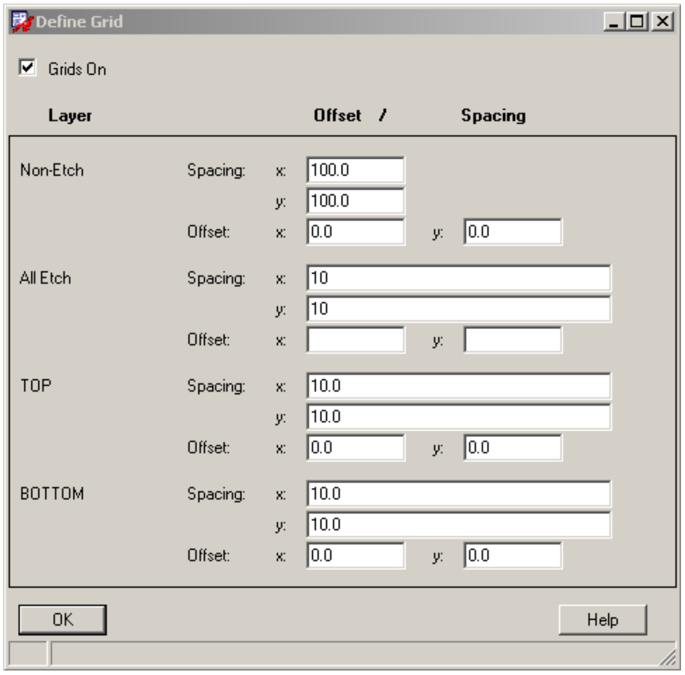
Board outlines can be drawn using lines or with a single rectangle, but if done with a rectangle, it *must* be selected from the *Add* toolbar, not the *Shapes* toolbar. We'll use lines instead of a rectangle, since we can set thickeness more easily, and use the menu sequence to avoid confusing it with the same icon on the Shapes toolbar. **Add → Line.** This will change the sub-mode (or sub-context) to "Add Line" causing the Options panel to reflect this fact:



Active Class and Subclass:
Board Geometry
Outline
Line lock: Line 45
Line width: 0.0
Line font: Solid

Notice the addition of the lock, width and font properties. They allow us to specify a solid line with orthogonal corners having a 25 mil width.

Notice as you move the mouse within the Design Window, the XY coordinates on the status bar move in 100 mil increments since this is the current Non-Etch resolution setting defined by the imported parameter file. Since our board is exactly 2.0 by 6.0 inches, this is an opportunity to show where the grid settings are done, and to change this setting temporarily to 1000 mils (1.0 inch) to make it easier to draw the board outline. Follow menu sequence **Setup → Grids…** to bring up the *Define Grid* dialog shown below.



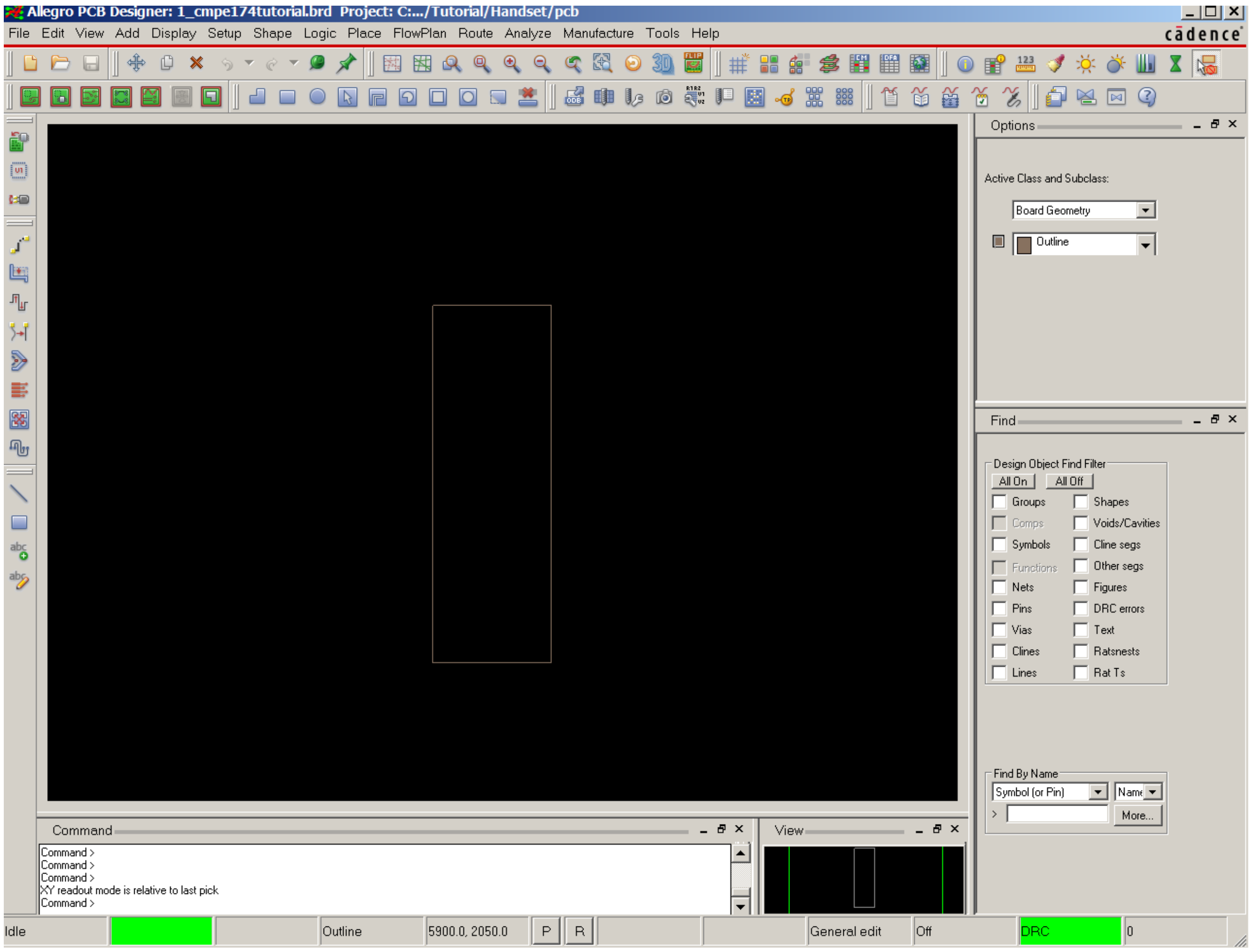
☑ Grids On

Layer			Offset /		Spacing
Non-Etch	Spacing:	x:	100.0		
		y:	100.0		
	Offset:	x:	0.0	y:	0.0
All Etch	Spacing:	x:	10		
		y:	10		
	Offset:	x:		y:	
TOP	Spacing:	x:	10.0		
		y:	10.0		
	Offset:	x:	0.0	y:	0.0
BOTTOM	Spacing:	x:	10.0		
		y:	10.0		
	Offset:	x:	0.0	y:	0.0

An additional parameter imported from the parameter file defined what basic units are used for the PCB design. I defned this to be *mils* (1 inch = 1000 mils); other options are inch, microns, millimeters or centimeters. Hence, the 100 you see in the *Non-Etch* fields calibrate equivalently to 0.1 inches. Change both the x and y spacings to 1000 mils.

The other three groups define the **routing grid** resolution and is currently at 1 mil for both the top and bottom copper layers. These can be changed collectively in the *All Etch* x and y fields. For example, making these 5 mils, will immediately propagate to both the Top and Bottom fields as well. Althought it's possible to set these parameters more finely than 1 mil, this is unrealistically precise. As you'll soon see, we'll change the routing grid later on to something more course than 1 mil. After changing the Non-Etch spacings to 1000 mils, press OK.

Before drawing the rectangle, zoom all the way out using the mouse wheel while looking at the green square in the *WorldView* window; it should be as large as possible. Now draw the rectangle somewhere in the middle of your drawing area. Watch the relative XY coordinates to confirm each side is the correct length as you proceed. After closing the rectangle, right-click the mouse to bring up the context-sensitive menu and select *Done*. If you did it correctly, the 2 by 6 inch rectangle should look something like the following:

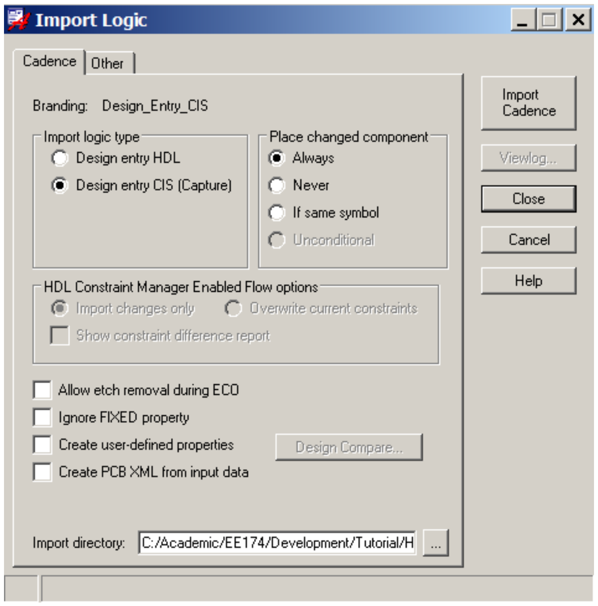


```
Command >
Command >
Command >
XY readout mode is relative to last pick
Command >
```

Don’t forget to go back and change the non-etch grid resolution to 100 mils before proceeding.

4.2 Importing the Netlist

Follow menu sequence **Import** → **Logic…** to bring up the *Import Logic* dialog. You shouldn't have to change anything on this panel. Be sure the Import Directory is pointing to your …Handset/pcb folder where the netlist files are located. The options in the *Place changed component* block are relevant only when we're updating a previously imported netlist. Press the *Import Cadence* button.



Branding: Design_Entry_CIS

Import logic type: Design entry HDL [✓ Design entry CIS (Capture)]

Place changed component: [✓ Always] Never If same symbol Unconditional

HDL Constraint Manager Enabled Flow options: [✓ Import changes only] Overwrite current constraints; ☐ Show constraint difference report

☐ Allow etch removal during ECO
☐ Ignore FIXED property
☐ Create user-defined properties
☐ Create PCB XML from input data

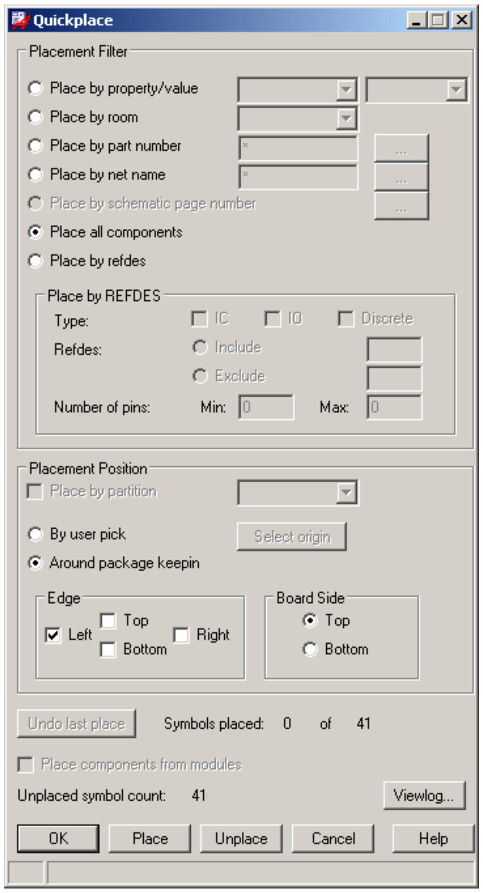
Import directory: C:/Academic/EE174/Development/Tutorial/H

If the import fails, you will need to carefully review the *Viewlog*. One is always generated whether successful or not, and can be viewed simply by re-opening the Import Logic dialog and selecting the *Viewlog…*button (it won't be greyed out like it is now, since no log existed when I took this screen shot).

Don't proceed to the next section until the netlist is fully imported successfully. The most common problem involves non-existent footprints due to naming errors back in Capture, or improperly specified library paths. Also, be sure your **\symbols** and **\padstacks** folders contain the right version of their contents. Currently this is ver 2.1. These can be downloaded from our website as single zip files. If necessary, download them and explode their contents into their respective folders.

4.3 Placing Imported Footprints

Placing footprints onto the visible PCB work area is very versatile in Allegro. This can be done automatically or manually. I'll show you both ways now.



Placement Filter: Place by property/value; Place by room; Place by part number; Place by net name; Place by schematic page number; [✓ Place all components]; Place by refdes

Place by REFDES — Type: ☐ IC ☐ IO ☐ Discrete; Refdes: Include Exclude; Number of pins: Min: 0 Max: 0

Placement Position: ☐ Place by partition; By user pick [✓ Around package keepin]

Edge: ☑ Left ☐ Top ☐ Right ☐ Bottom

Board Side: [✓ Top] Bottom

Symbols placed: 0 of 41

☐ Place components from modules

Unplaced symbol count: 41

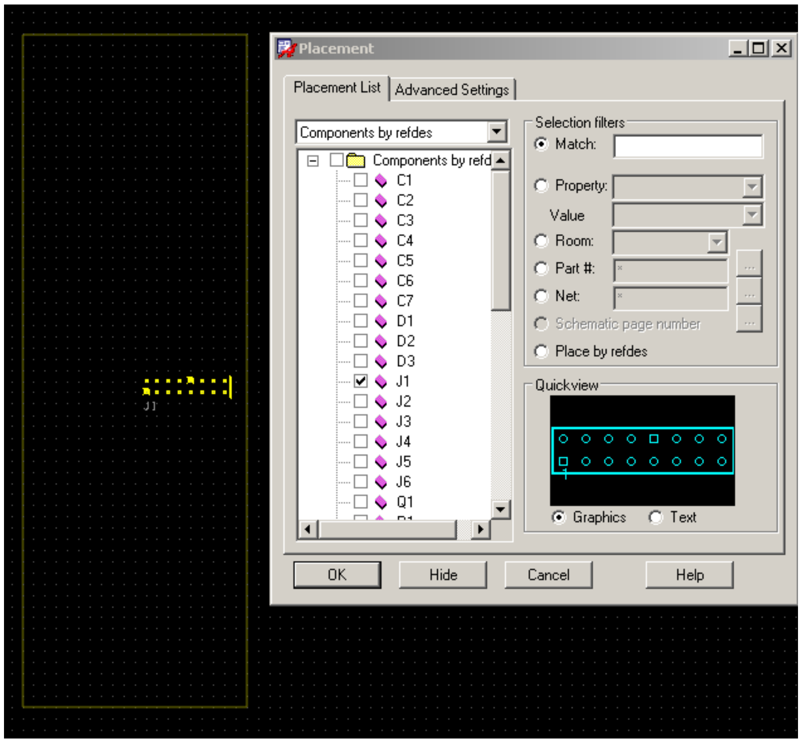
Automatic Placement – Choose **Place** → **Quickplace…** to bring up the *Quickplace* dialog. Typically, this is the fastest way to get parts on the board, but pre-placement may not be to your liking. Notice the *Unplaced symbol count* at the bottom. This tells us how many footprints have not yet been placed, in this case 41. Had we placed some of them manually, then this number would be less by that amount. I have set the fields to place new footprints along the left edge of the board outline and on the top side of the board (same side as Top Copper).

Pressing the *Place* button puts them on the board, *Unplace* removes them! Try it. Experiment with the edges if you want to see where Quickplace actually puts them. Note that if you close this dialog (by pressing OK) and then re-open it, the Unplace feature no longer works; only items "placed" when the dialog is open can then be "unplaced". For our purposes, I'll assume everything has been placed along the left edge. Status messages will appear at the bottom of the form. If all footprints were placed, the symbol count should be 0.

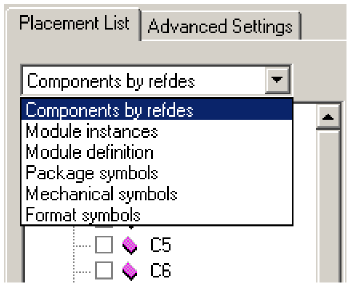
Use either the *all_layers* or AST color views available in the Views' pull-down list on the Visibility panel. These two color views include some indication of what the part's package looks like. I used the AST view while doing initial importing.

Manual Placement

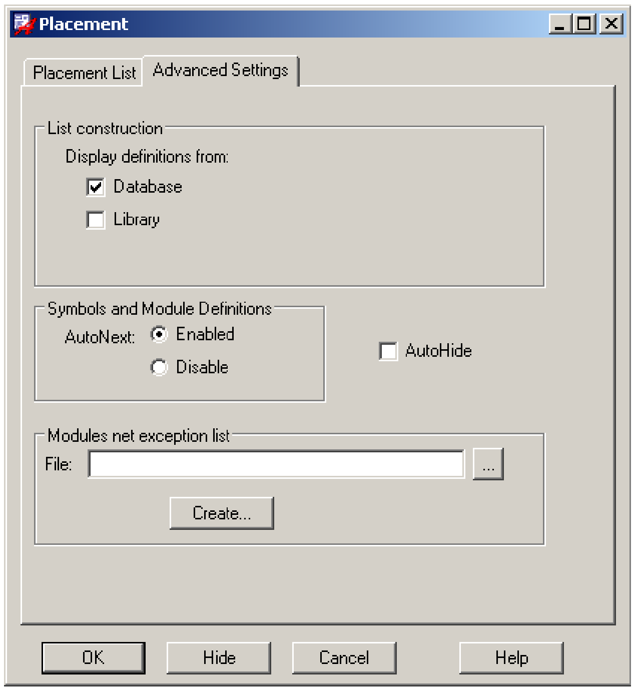
Choose **Place → Manually…** to bring up the *Placement* dialog.



Match: [✓ Match:]
J1: [✓ J1]
Quickview: [✓ Graphics] Text



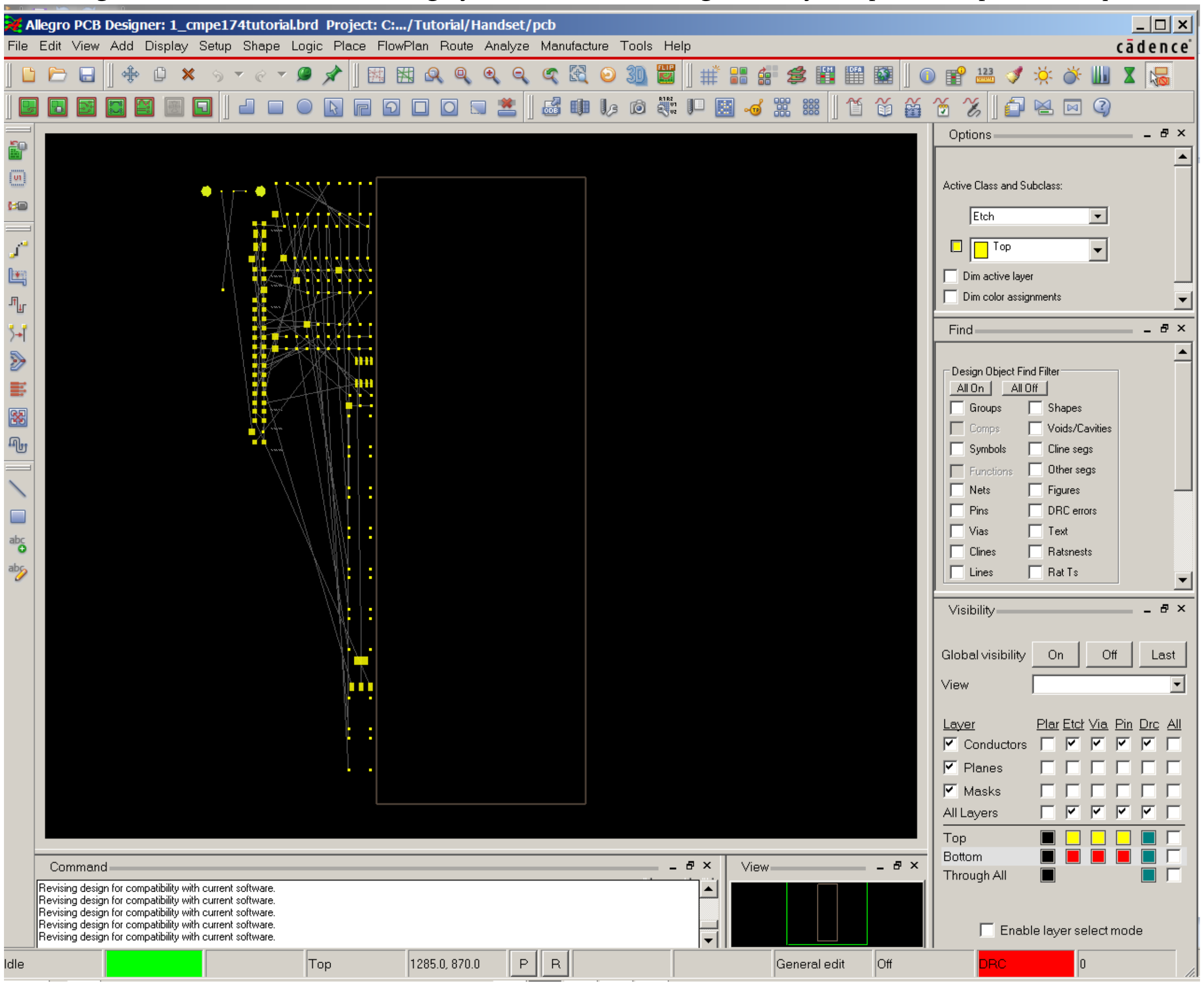
Since we want to place footprints from our loaded netlist, we choose *Components by refdes* in the pull-down list, and on the *Advanced Settings* tab, we specify how the list of components is to be constructed. As shown, I checked reference designator J1 and immediately moved the mouse onto the work area, as shown (don't press OK, it will just close the window). After positioning the footprint, the left mouse button places it on the PCB and Allegro flags the entry with a cute icon replacement for the oblique purple rectangle. If you then close the dialog and reopen it, any placed parts won't be present in the list. If you check everything in the list, and *Autonext* is enabled as shown below, then you can quickly pre-stage everything (try it).



Database: [✓ Database]
Library: [ ]
AutoNext: [✓ Enabled] Disable
AutoHide: [ ]

On the **Advanced Settings** tab, the *List construction* block tells Allegro what libraries we want to include in the list above. Definitions can be taken from "Database" (Cadence-speak for only the loaded netlist footprints) or "Library", which includes everything from the master footprint folders specified in *psmpath*. Since we only want the loaded netlist parts that came over from Capture, Database is the only one checked (this is also the default).

Manual placement allows footprints to be placed further apart than if otherwise done automatically. Whether you place them using Quickplace or do it manually as I've done, your design should now look something like the following screen shot. The **ratsnest** of grey traces interconnecting the many components represent unique named



Active Class and Subclass: Etch, Top

```
Revising design for compatibility with current software.
Revising design for compatibility with current software.
Revising design for compatibility with current software.
Revising design for compatibility with current software.
Revising design for compatibility with current software.
```

Idle | Top | 1285.0, 870.0 | P | R | General edit | Off | DRC | 0

nets. Each one will be used during the process of *routing* to convert them to copper tracks causing them to disappear! The purpose of the ratsnest is to enforce connections only allowed by the master netlist brought over from Capture, and to remind us to route them. Visibility of the ratsnest is easily controlled by the following two icons:

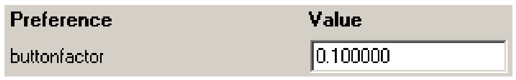
Turns off the ratsnest, making it invisible.

Turns the ratsnest on, making it visible.

When placing parts, the ratsnest is a nuisance, so just turn it off to unclutter the screen (but it will need to be on for routing). Turn off the ratsnest now.

Zooming and Panning

If you haven't figured out already how to zoom and pan around in the Design workspace, the center mouse wheel zooms in and out *wherever the mouse is currently pointing.* This is one way to "pan": zoom out, point, zoom back in; the other way is to press the center mouse wheel (hold it down) in a classic click-and-drag fashion to rapidly pan about. You will find it useful to change the mouse wheel's zoom ratio to something less dramatic – currently it's set for 2x which is quite large. This value is defined on the *User Preferences Editor* dialog in the *Ui/Zoom* Category.



Preference	Value
buttonfactor	0.100000

Set the *buttonfactor* to 0.1 instead of the default 1 (the actual zoom factor is twice this entered value). [5]

[5] See page 10 if you forgot how to access the User Preference Editor.

4.4 Placing Mounting Holes

We'll begin by first defining the six mounting holes. Refer to the fully dimensioned drawing (see Appendix-C) to visually see where these go and determine their coordinates. Since you're new at this, I've included a brief checklist quickly noting how the visible workspace should first be configured:

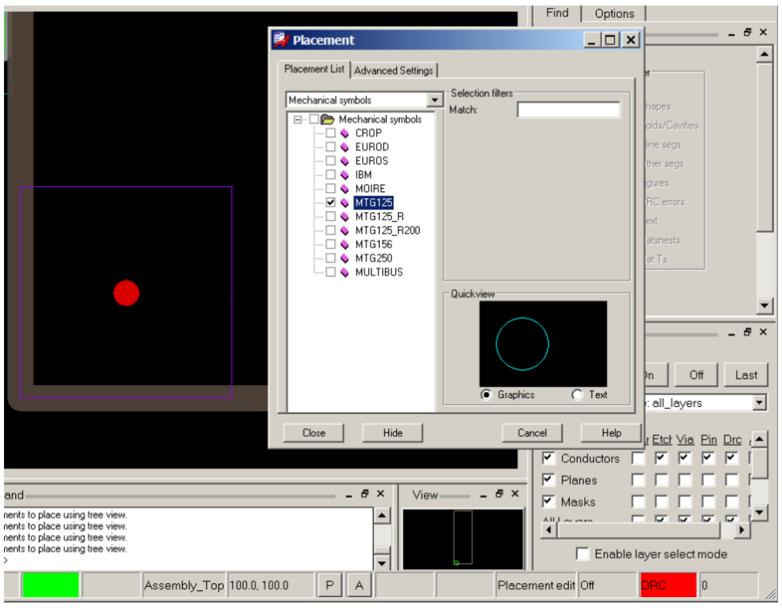
1. The *Drawing Origin* should be defined at the lower left corner of the board outline.
2. Non-etch grid is set to 100 mils and grid is visible (use the *Grid Toggle* icon).
3. Master mode is *General*.
4. XY mode should be *Relative*.
5. Ratsnest should be off.
6. View should be *all_layers*.
7. Set *Global Transparency* to 100% to make everything vividly bright. We'll change this later when actually routing.

Setting the Drawing Origin

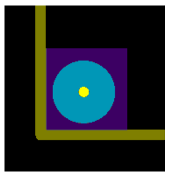
The XY coordinates as you move the mouse around are always displayed at the current grid resolution on the status bar. This display has two mode: *relative* and *absolute*. Relative shows delta X and delta Y relative to the last mouse click, while absolute is the actual XY coordinates. The *drawing origin* is the master absolute X=0, Y=0 drawing reference. Set it now to 0,0 by first setting setting the display for absolute coordinates by making sure the XY Mode icon displays "A" at the bottom of screen on the status bar. If it reads "R", it's in relative mode. With the grid visible in Placement Mode, zoom in on the lower left corner of the board outline and choose **Setup → Change Drawing Origin** and click on the lower left corner. Observe the XY annunciator fields in the status bar now show "0.0, 0.0".

With the absolute origin now set and the grid turned on so it can be seen, change to relative coordinates and zoom in on the bottom edge of the board outline until the 100 mil grid can be cleary seen. Remember, Cadence changes the grid's apparent visual spacing if the view is too large, so we want to be sure we're seeing the wanted grid setting of 100mil. Access the manual placement menu and select *Mechanical symbols* from the "Library" (checked on the Advanced Settings panel). Now, here's how it works. We want to place these holes in a particular location; we know the two on the bottom are exactly 100mils in from their respective corners. The technique is simple: first, move the mouse to one of the corners as exactly as you can visually and click the left mouse button. If this point is close enough, Allegro will snap to the nearest grid point "setting" it as the referencepoint, and that point should be the one you want. Now select the 1/8 inch mounting hole, "MTG125" (125mils = 1/8inch) and move it down to relative XY coordinates = 100,100. The following screen shot shows this positioned *but not yet placed*, *i.e.* I haven't yet clicked the left mouse button.

A few comments before placing it. Since all layers are visible, the purple square you see is a special drawing telling Allegro's auto-router (and alert you too) not to route any traces inside it. It is defined when the part was created and appears on a special logical layer in the Class / Sub-class: *Route Keepout / All*. The red dot is a very small pad on the top copper layer (Etch/top) indicating where the drill center is located. Finally, notice the relative XY coordinates are 100,100 as they should be.



After placing the part, its appearance will change.The new larger aqua-colored filled round area is a 125mil pad on the top and bottom soldermask layers the same size as the actual drill-size; you can also see the route keepin area now appears as a filled object. Changing the View to either AST or

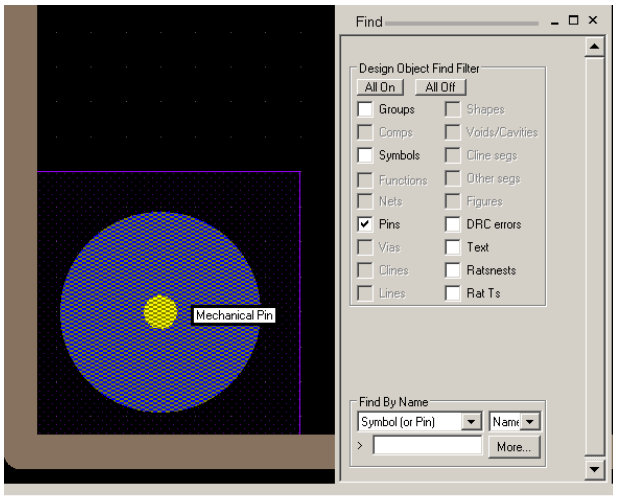


ASB removes the keepout, but shows the top copper and soldermask layers. These are the layers we want visible while routing traces, as you'll see shortly.

Note: Once a symbol is placed, it will disappear from the list in the Placement dialog, but only after closing and reopening it. Prior to this it can be repeatedly placed from the list. Adding a part that is already present will then require copying one that has been already placed.

Using the Find Panel

This is a good opportunity to discuss how *filters* work on the **Find** panel. Eventually, when your PCB is covered with objects: traces, text, pins, footprints, vias etc., and you want to work specifically with only one or a sub-set of them, like editing this mounting hole for example, selection filters can be judiciously used for only particular objects of interest. Knowing how to use these filters is absolutely essential to skillfully working with Allegro.

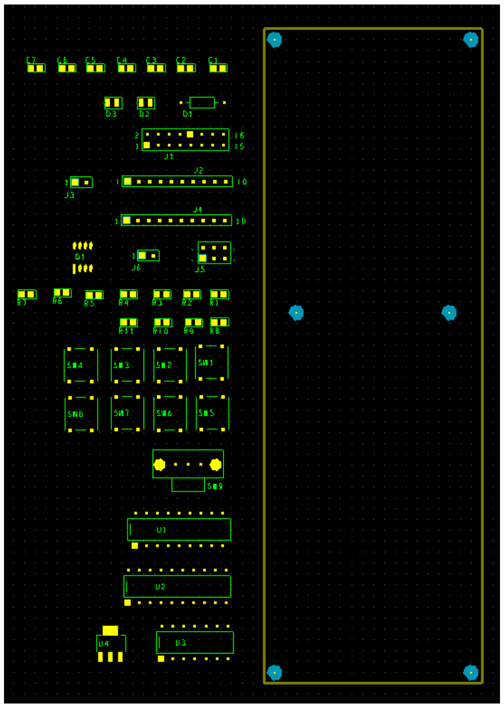


As an exercise, experimentally determine what kind of object Allegro assigns to mounting holes. Do this by first pressing the *All Off* button to quickly uncheck everything, then activate each filter type and hover the mouse over the hole to see of Allegro recognizes it. I've done exactly that and found that Allegro assigns holes to *Pins*; in other words, it's a pin object, and Allegro proceeds to tell us what *kind* of pin it is: a "Mechanical Pin". This is important, since the other kind is a *Connect* pin and will always appear in the netlist, making it an electrical object rather than a mundane mechanical object.

Later you'll find selecting certain objects will only be possible with proper filtering to enforce exclusion of incorrect ones that may overlay or be nearby. Which set of possible objects appears is also dependent on what master mode and particular sub-modes Allegro happen to be in. In other words, the list is context-sensitive; note above that certain items are greyed out because the current context doesn't use them.

TIP: A quick way to measure distances between any two grid points is to first turn off all filters and simply click the mouse one of the two points of interest. All mouse movements will then be relative to this "anchor" point; but, this only works in XY relative mode. For example, try this now by confirming your board really is 2 by 6 inches.

Continue to place the remaining five mounting holes. Below I show the result of this effort (AST view).



One last comment. Why do you think the yellow copper pad for each of the six holes on the top layer is so much *smaller* than the surrounding solder mask and, most to the point, the actual drill size? The reason is simple, we want this to be a very plain 125mil drilled-only set of holes having no plating (pads). By making the plating much smaller than the drill, it will disappear when the hole is actually made!

4.5 Standards of Good Engineering Practice

With the mounting holes in place, we're ready to place parts and begin routing traces. Placing parts should always be done with a copy of the latest applicable version of the corresponding schematic on hand. You may also have layout constraints noted from app notes and data sheets; these of course, would be found in your engineering notebook. Here, you should refer to both the schematic and example layout pages provided with this lab. I will be using the instance designator references from the first page showing the board dimensions (you may need to refer to the original handout schematic from Capture to correlate these with your board – especially if you changed any of the reference designators during the annotation process). If you did, simply note these in pencil on your working schematic. Keep this document properly filed in your engineering notebook – not as a free loose-leaf note that can easily end up in your English homework.

Placing and wiring parts effectively requires that we carefully consider relevant electrical issues involved. First, it helps considerably to classify traces as being associated with either **Power** or **Signals.** Only *power traces* route voltages and their ground returns carrying considerable current to chips so they can be powered. *Signal traces* don't carry power, so they lack the associated heavy current demands; rather, they carry voltages representing analog or digital information between devices (outputs to one or more inputs). Note that in high-speed circuits, signal traces may carry very high transient currents, but these will always average to essentially zero over time; power traces will also carry these associated transients but they won't average to zero.

At the board level, then, we have two basic types of routing: *power traces* and *signal traces*. It is absolutely essential that you understand the difference between them as they are handled quite differently while laying out any board. These differences are summarized in the following list of board layout guidelines, considered as *standards of good engineering practice*:

1. **Place all parts to minimize associated inter-device wiring distances**. In other words, some parts go naturally next to each other since they have interconnected traces. This applies primarily to signal traces, since power traces may be treated differently as noted following.

2. **Always establish low impedance power and ground nodes**. This will generally insure that current demands on the power traces won't cause the supply voltage to vary. The lab design uses an LM317 linear regulator to produce, ideally, a constant +4.5V. The impedance looking "back into" the LM317 between its output and ground pins is very low by design (this is also the same as the Thevinin output impedance of an ideal voltage source). In designs where current transients demanded from the regulator are high, you should also place a "bypass" or "filter" capacitor across the regulator output as close to the output and ground pins as possible. When the regulator cannot respond to very rapid current demands (this is a magnitude issue; it's a speed issue), the cap will supply or sink the transient current and so stabilize the voltage across it. Linear regulators stabilize their output voltage using internal feedback control. Bypass capacitors can destabilize this by changing the phase margin. *Always refer to relevant datasheets and app notes when determining how to use any particular device*. This is the reason your engineering notebook has space specifically for such Appnotes and Datasheets!

3. **Power and ground traces require special attention**. They should be routed using wide, low inductance traces to minimize unwanted voltage variations at power pins due to rapidly changing current demands running through these necessarily inductive traces.

4. **Route power and ground traces to minimize inter-chip coupling through the power pins**. This topic is somewhat involved. See my paper, ***Power Distribution and Part Placement Note***, esp. section 1.2 discussing inter-chip coupling (note: this would be an example of a reference to place in your Appnotes folder section).

5. **Place bypass capacitors close to the pins needing "bypassing"**. Bypass caps are included by design engineers specifically to stabilize unwanted power supply variations caused by the unavoidable (and hence, "parasitic") inductance associated with ground and power traces noted in guideline 3 above; see the paper noted in 4 above for a complete discussion of this topic. They work by forming a low-pass filter with the power supply's positive and negative (typically "ground") leads' parasitic inductances. An

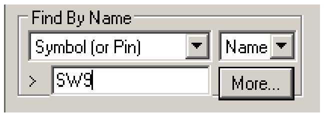
example is U3, pin-14 on the 74HC00 quad nand chip. It follows that since trace inductance varies inversely (and quite non-linearly) with trace width, making traces as wide as practical is good. In some cases, we pay so much attention to this that we may devote entire inner board layers to just power and ground routing! As signal frequencies increase this consideration becomes more critical. Four-layer PCB's are often predicated on this goal, where the two outer layers are meant for parts placement and signal routing, while the two remaining inner layers are defined as **power planes**, and will exhibit very low routing inductance. This topic is discussed in cmpe173, High Speed Digital Design. Two-layer boards often realize a compromise by adding low-impedance **copper pours** or **areas** on either or both the top and bottom layers assigned to carry power.

4.6 Placing Footprints and Manually Routing Nets

So let's begin! Using General Edit mode [GEN], set the selection filter for only *Symbols* and activate the *Move* context by either menu sequence **Edit ⟶ Move** or with the icon shown. The only indication of this new sub-mode is the Options panel's content will change. Simply select a footprint with the mouse and move onto your new board, using the views found in Appendix-C as a guide. At this point you don't need to be precise. Parts can be rotated in 90 degree increments using the context-sensitive submenu *rotate*. You might notice, if you zoom in closely, the grid has changed from the 100 mil non-etch to whatever the **All Etch** grid is; set this to 50 mils (both X and Y) before proceeding.

Since we're working only with footprints, a better way to work is to change to Placement Edit master mode [PLC]. Do so now. In this mode we don't need to explicitly activate Move mode, but be sure to set the filter to *symbols*. Note too the other objects you can work with in the Find panel.

I like to begin by aggregating parts based on the engineering schematic. For example, "clump" all the passive components associated with U3, the 3Hz oscillator and 10us pulse generator together; this would include U3, C2, R4, R2, R3, C3 and C7.



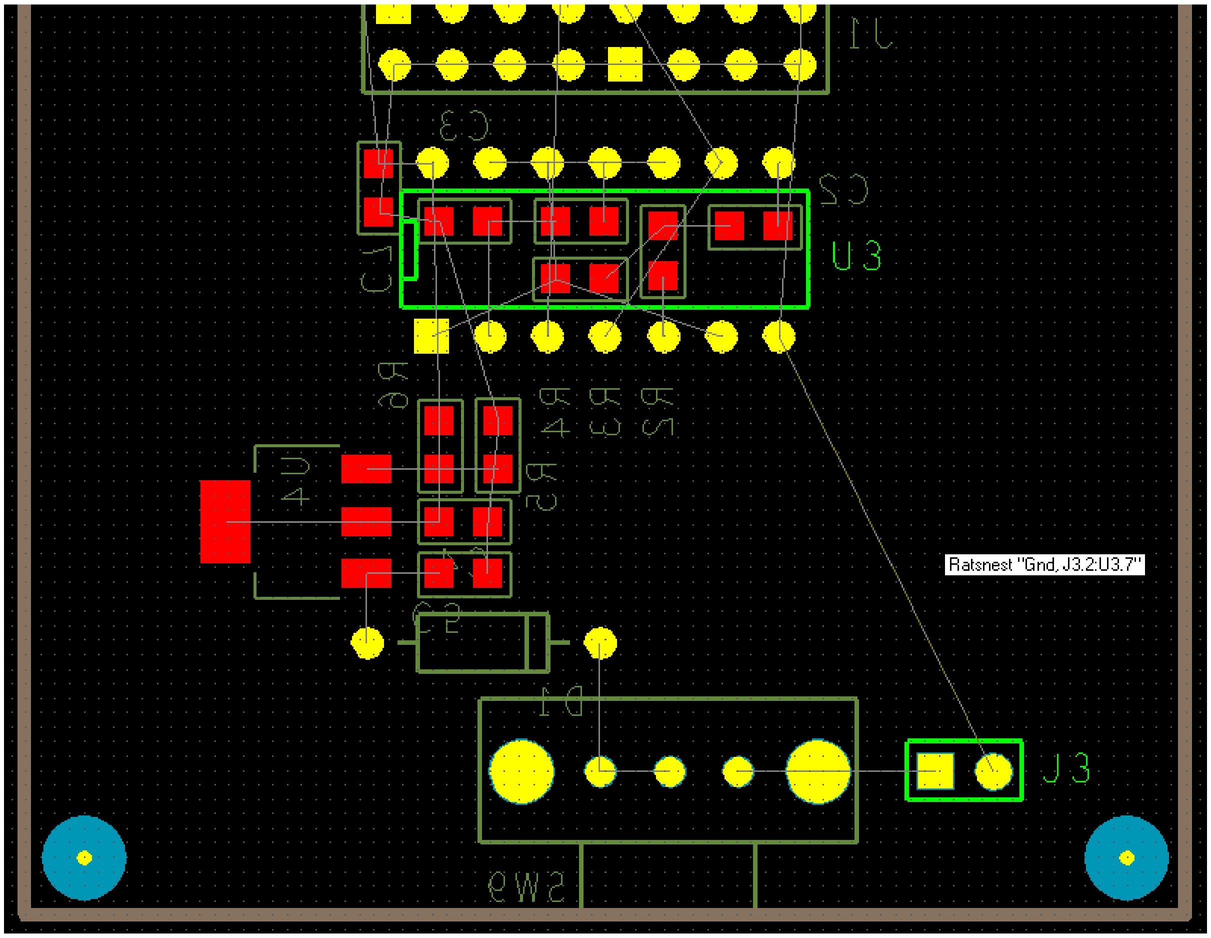
Locate parts by using the *Find By Name* options at the bottom of the Find panel. For example, you might want to locate SW9 as shown. What type of object you're looking for is specified in the list-box; as shown, we're looking for *Symbols* (footprints) or *Pins*. Pressing *More* brings up a complex dialog whose use should be self-evident.

After finishing initial placements, you may need to make the appropriate grid finer to precisely place parts as shown in the handout, but <u>don't make it smaller than 5 mils</u>. Position the buttons and DIP's using 50mils, and the small SM parts with 5 or 10mils. You should also turn on the ratsnest to enable part placements that intelligently minimize inter-device wiring lengths, particularly with respect to rotating them into correct position. While it is true that in this design you already have a guideline to go by, understand that none of these parts were placed without first carefully considering their purpose in light of the standards of good engineering practice. You should confirm this too, endeavoring to understanding *why* they're placed as they are. After getting parts "clumped", go ahead and place them as accurately as you can visually based on the handout. Note that many surface mount parts are located on the bottom of the board. This was done to protect these parts from finger access on the top. Any part can be moved to either layer by hovering over it (but not actually selecting it), and choosing *Mirror* from the lengthy right-click context-sensitive menu. Proper rotation and mirroring will take place automatically since the top and bottom layers are defined to be "mirrors" of each other. Placing SM ("surface-mount") parts beneath through-hole DIP parts is a very common and useful technique, since we can take advantage of the via between the two layers to easily connect to pins on the DIP's footprint.

Routing Nets

First, turn the ratsnest on. You should see something similar to the following screen shot. Always remember that Allegro draws the ratsnest based on the *closest connection points having the same net name*, not necessarily *where* the trace is actually supposed to go. This feature can best be seen with the GND and +4.5V nets, since they will have a larger number of connected nodes. Try moving one of the parts to observe this. The point of this exercise is to show you that the ratsnest merely displays connections to nearest nodes having the same connectivity, not necessarily which nodes (or pins, say) that must actually have particular traces routed between them. Eventually, of course, all the pins connected to GND will in fact be connected together (since they have the same connectivity), but not in the particular way the ratsnest might otherwise be suggesting this. Beginners tend to overlook this and slavishly try to "follow the ratsnest". Don't be mislead!

The parts shown in red are those I placed on the bottom layer, while those in yellow are on the top since these colors are assigned to the top and bottom etch sub-classes respectively. If your board doesn't show only red and yellow



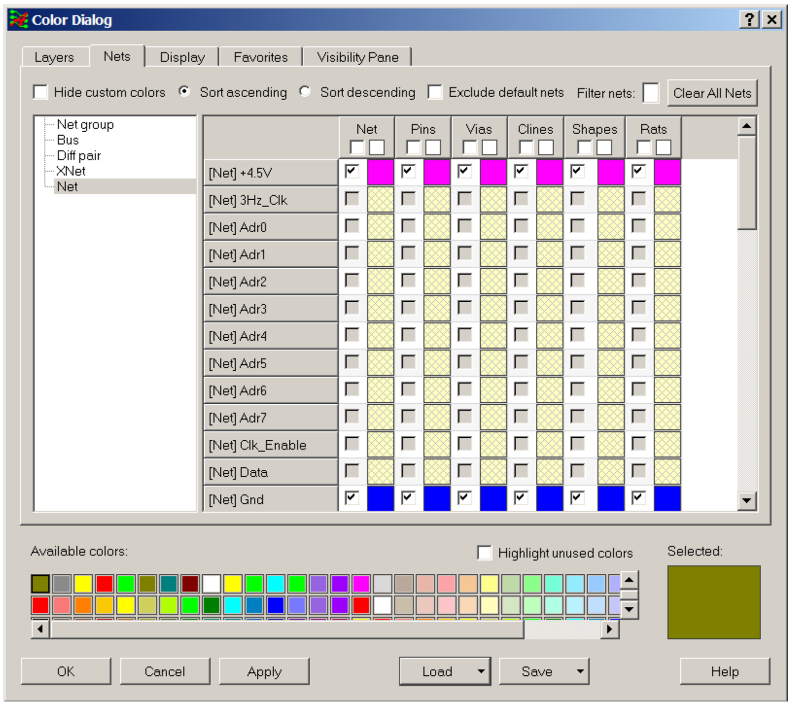
Ratsnest "Gnd, J3.2:U3.7"

We will be routing power (+4.5V net) on the top layer and most of the GND connections directly to a large copper plane on the bottom. Any through-hole pin tied to a GND net will automatically get connected to this copper plane after we create it. I will show you how to do this near the end of the lab. For now, how do we tell which nets are GND or +4.5V and which aren't so we won't try and route traces to them? One way is to hover the mouse over any rat line and let Allegro tell us what it is. I've done this with one of the two nets connected to J3; the rat line "highlighted" into a segmented appearance and a text message appeared nearby telling us it's part of the GND net.

Try it. If the rat line doesn't highlight, change master mode to Etch Edit [EE] and enable only *Ratsnests* in the Find filter. Then as you move the mouse about only rat lines can be selected. But there is a much better way to distinguish a large set of nets, like GND and +4.5V, and that is to color-code them.

Color Coding Nets

Bring up the **Color Dialog** and check the *Nets* tab. Every net in the design will now be available for color coding.



☐ Hide custom colors [✓ Sort ascending] Sort descending ☐ Exclude default nets Filter nets: ☐ Clear All Nets

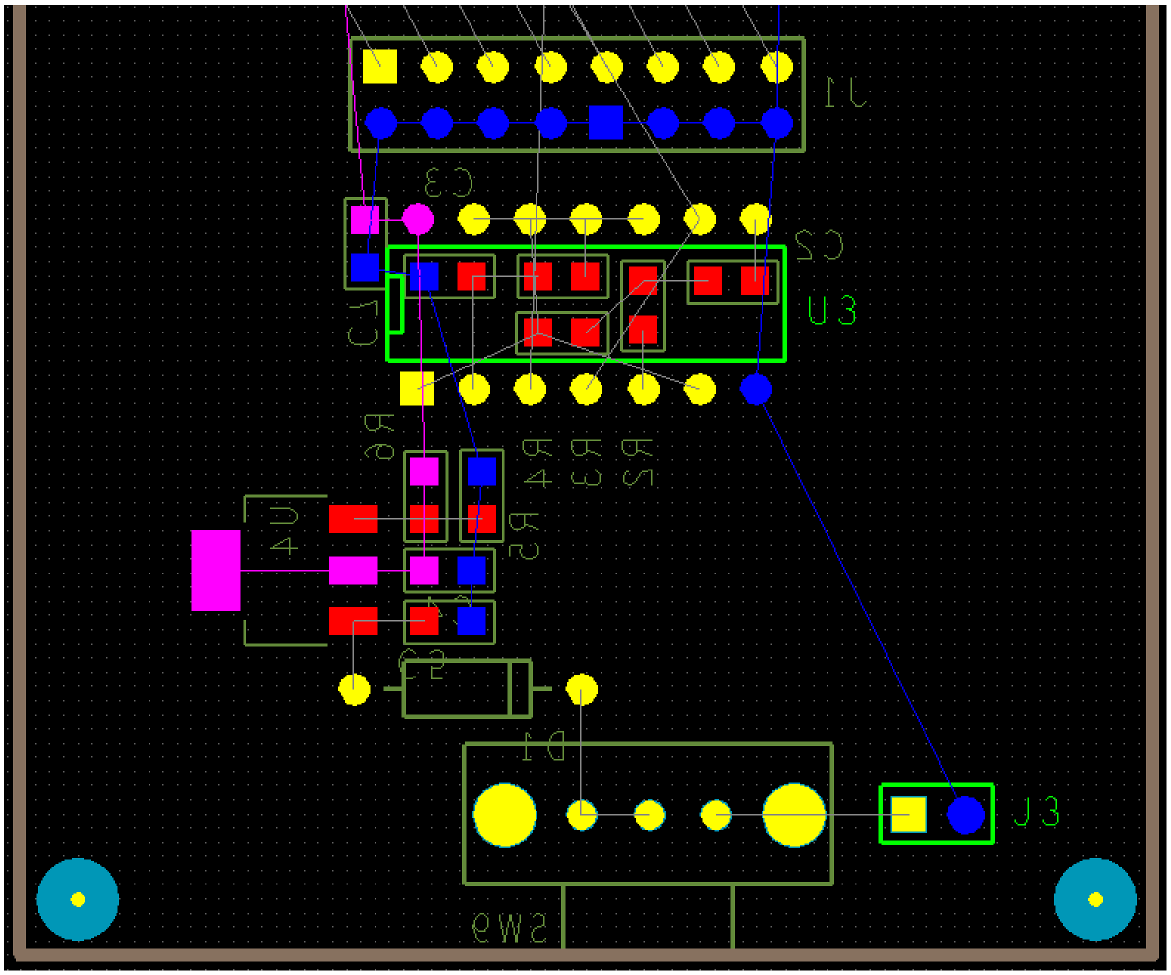
	Net		Pins		Vias		Clines		Shapes		Rats	
[Net] +4.5V	✓	🟪	✓	🟪	✓	🟪	✓	🟪	✓	🟪	✓	🟪
[Net] 3Hz_Clk	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr0	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr1	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr2	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr3	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr4	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr5	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr6	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Adr7	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Clk_Enable	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Data	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨	☐	▨
[Net] Gnd	✓	🟦	✓	🟦	✓	🟦	✓	🟦	✓	🟦	✓	🟦

Available colors: ☐ Highlight unused colors Selected:

Assign blue to GND and violet (but not red) to +4.5V as shown. Note that we can do more than just change a rat line's color, we can also change other object's associated with the GND net. Here, I've given them all the same color and made them all visible. What happens to the workspace is shown in the next screen-shot.

Every object associated with the GND net now appears blue; similarly for the for +4.5V net. This will make it easy to exclude any through-hole GND pin or grounded SM part from our routing chores.

Check the *Hide custom colors* checkbox to revert to uniform layer definition colors. When confirming whether an object is on the top or bottom layers, this feature *must be turned off!*

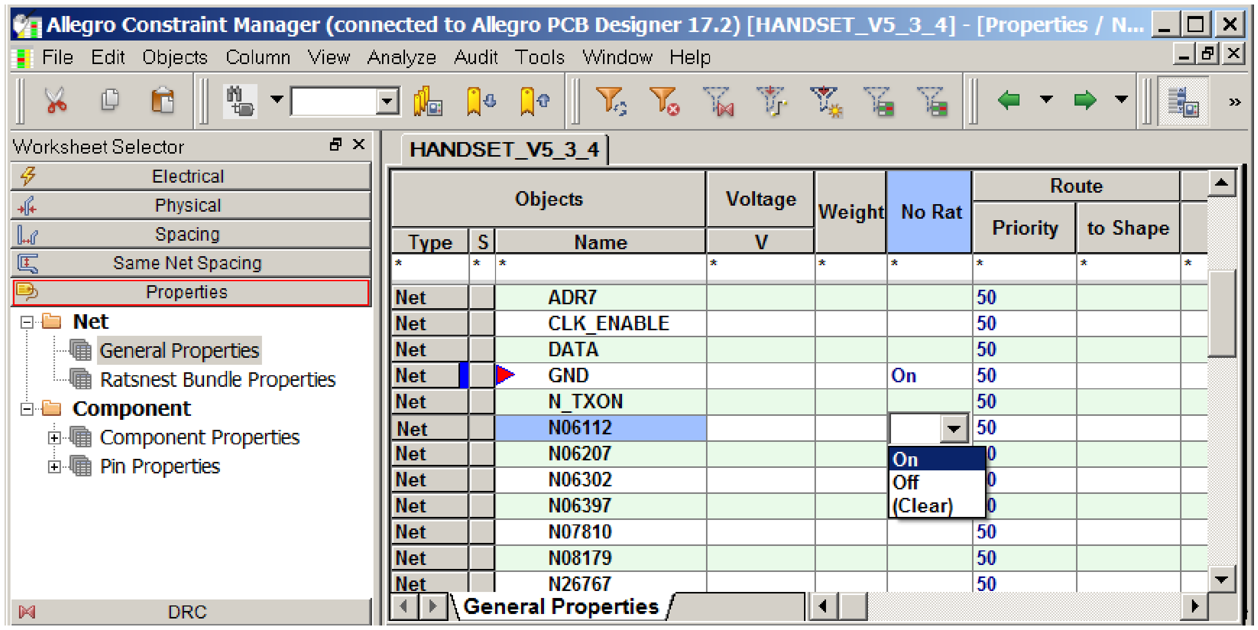


Selectively Turning Off Rat Nets

Sometimes it makes things less cluttered by making certain nets invisible in the rats nest. A good example is the ground net since much of the connectivity will be through a copper plane on the bottom layer. This feature is accessed through the *Constraint Manager* icon .

The constraint manager is quite complex. Shown below is the section dealing with Nets and General Properties. The *No Rat* field can be clicked directly to access a pull-down menu with three options shown for net N06112 as an example:

On: makes the ratnest for that net invisible.
Off or (Clear): makes that net's ratnest visible again.


Allegro Constraint Manager (connected to Allegro PCB Designer 17.2) [HANDSET_V5_3_4] - [Properties / N...
File Edit Objects Column View Analyze Audit Tools Window Help
Worksheet Selector
Electrical
Physical
Spacing
Same Net Spacing
Properties
Net
General Properties
Ratsnest Bundle Properties
Component
Component Properties
Pin Properties
DRC
HANDSET_V5_3_4
General Properties

Objects			Voltage	Weight	No Rat	Route	
Type	S	Name	V			Priority	to Shape
*	*	*	*	*	*	*	*
Net		ADR7				50	
Net		CLK_ENABLE				50	
Net		DATA				50	
Net		GND			On	50	
Net		N_TXON				50	
Net		N06112				50	
Net		N06207			On	0	
Net		N06302			Off	0	
Net		N06397			(Clear)	0	
Net		N07810				50	
Net		N08179				50	
Net		N26767				50	

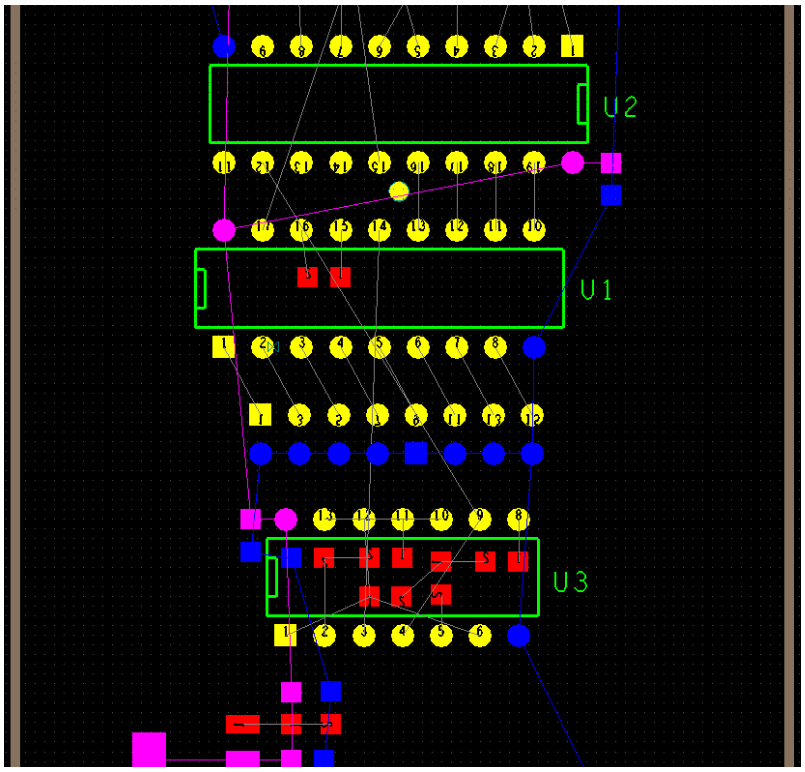
DRC Settings

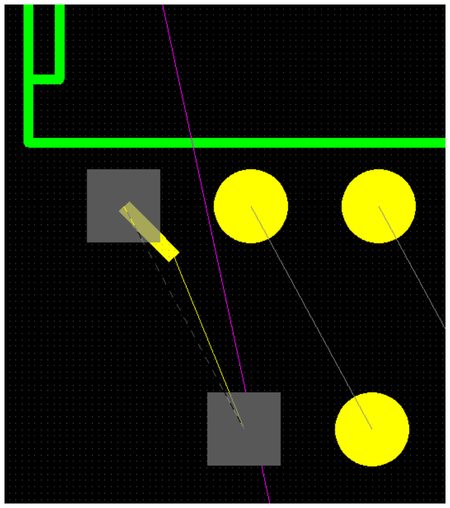
Before manually routing be sure to enable dynamic drc checking. Follow menu sequence:
Setup⟶ Enable On-line DRC… Beginners should always have the Online-DRC enabled while routing. After you get the hang of things it can be turned on and off as you see fit. It's particularly useful for automatically making sure no spacing violations occur.

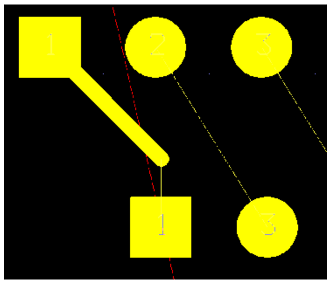
Refer to **Appendix-C** or **…AppNotes\Lab2 Addendum Views rev 1.1.pdf** for several views of the handset showing routed nets on the top and bottom layers.

With all net visible proceed to begin routing!

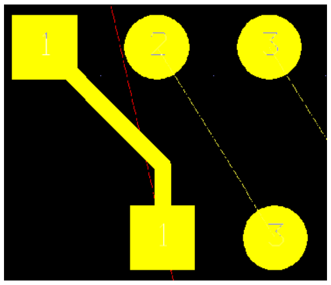
Set Etchedit mode, the Find filter for only *Ratsnests* and the colorview to AST. Begin manually routing some nets. Route the easy, obvious one's first: U1 pins 1 through 8, for example. Although these are on the bottom layer, we'll first practice routing them on the top. Whenever the mouse is clicked on a net, Allegro automatically assumes you want to begin the trace from the pin *nearest to where you selected the ratsnest*. Try this with U1-1 two ways: click close to pin-1. The ratsnest will divide into two segments (as shown above): the thick yellow line represents the new routed trace *from* the pin it starts on, and highlights yellow for the remainder of the net yet to be routed (if the angle is 90 degrees instead of 45, set the *Corners:* to be 45 in the pull-down list that appears in the *Options* dialog whenever routing is active) .





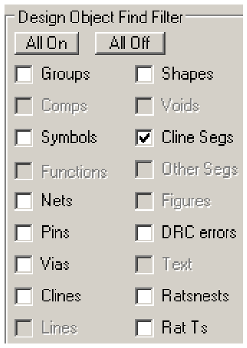


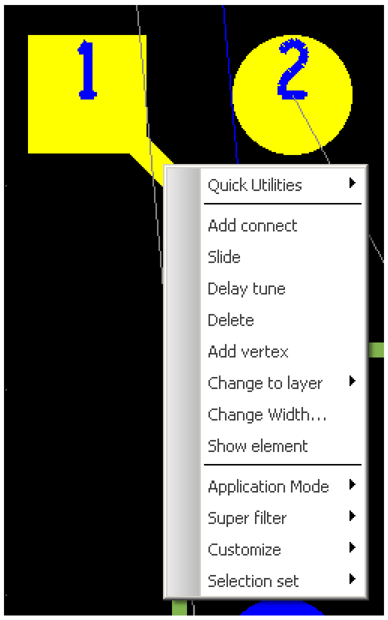
Now move the vertex point just above J1-1 and click the left mouse button. This will route the *first segment* from U1-1 as shown. Continue moving the mouse *closer* to the wanted termination pin, J1-1, and click the left mouse button when you're right on the center of the pin.



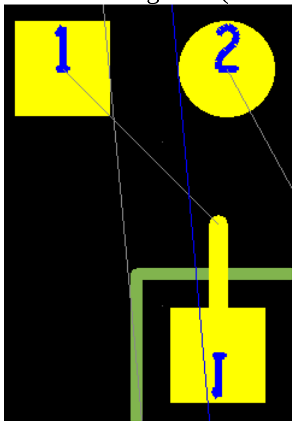
Allegro will route this *second segment* just as it did the first one, beginning from the last routed vertex point (at the bend) and terminating on J1-1. If the mouse had been closer to the vertex point instead of the wanted ending point at J1-1, Layout would have simply added another short segment track from the vertex to where you clicked the mouse.

You should now remove this track and try routing it beginning from J1-1 instead, but first it needs to be removed. The process of removing a track or net is called **ripping**. Two ways exist to do this: one segment at a time or the entire connected trace. I'll show you both ways now. Cadence calls tracks or traces *connected lines*, abbreviated *Cline*. We'll do the segments first. Set the object filter to work only with connected line segments by checking *Cline Segs*.

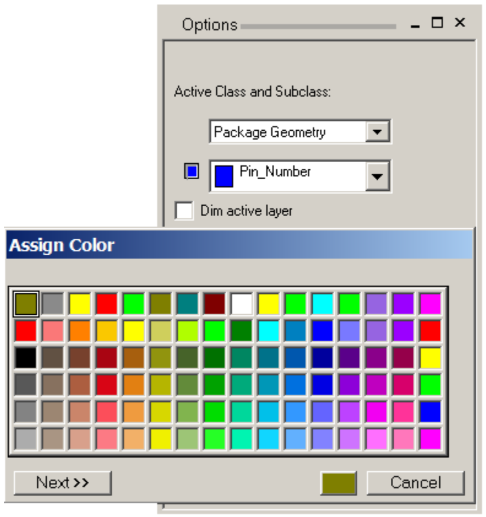




Move your mouse over the two segments and observe how each of them alternately highlight. Hover over either, but rather than select it, access the context menu (right-click the mouse) shown. Choosing *Delete* will now remove it from the track (try it.). Repeat to remove the second segment (shown below).



By the way, I turned on the pin numbers so you can see them and changed their color to dark blue. As an aside, why do you think J1-1 appears upside down (think about it)?



CHANGING COLORS IN THE OPTIONS PANEL – Seeing pin numbers is a good example to show how this is done. Select *Package Geometry / Pin_Number* (in the Options panel) turn on the layer as it will most likely be off, but the default color is aqua and doesn't have much contrast against the yellow pads. Right click the little color square to the left of the sub-class string entry and select a suitable color – dark blue in this case (try it). This change is permanent, so change it back unless you like this new color (obviously the board file must be saved first…). Feel free to set colors to suit yourself. Later, if you really like them, they can be saved in a custom parameter file.
For now, you'll just have to put up with mine!

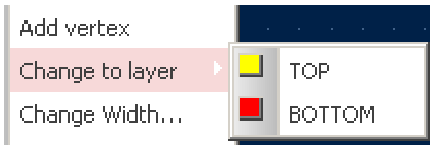
Reverse the two previous delete operations by undoing them twice (use CTRL+Z or **Edit**⟶ **Undo** or icon). Now select *Clines* only and repeat the operation. The entire track now highlights as Allegro recognizes it; hover, right-click, choose *Delete* and the whole connection is gone.

Tip: Control-Z sometimes fails to work. If this happens, make sure the Command line isn't clogged with nonsense. Clearing it usually fixes things.

You have just worked with two filters: *Cline Segs* and *Clines*. Once a track exists, any manual change must be based on one of these two. Hence, it's important to understand what they're capable of so you can intelligently choose one or the other; if you happen to choose both, Allegro defaults to Cline segments. Carefully comparing the two sub-menus reveals they have some options in common, like delete, but some that are also unique. Understanding these differences is essential to effortlessly editing tracks. Since they both overlap (delete for example) it's easy to choose the wrong one and try and make it do something that only the other is capable of. The following summarizes use of these modes, and should be used as a reference.

Clines and Cline Segs Context menu Options:

- **Add connect.** Segments only. This is equivalent to a highlight-and-select operation with the mouse.
- **Slide.** Segments only. This is a very useful command. Existing routes can be moved about in segment mode and repositioned by holding the selecting a segment and holding the mouse down in a click-and-drag like operation. Selecting this option releases you from having to hold the mouse key down, and allows us to change the many slide settings that appear in the Options panel.
- **Delay tune.** Segments only.
- **Delete.** Segments and Clines
- **Add Vertex.** Segments only. This will break a segment in two at the hover point of the mouse. This is a fast way to add "some bends" (more segments) to an otherwise straight line section.
- **Change to layer.** Segments and Clines. Provides a means to change the selected object (segment or line) to another etch layer. In our design, this would be top or bottom as shown.

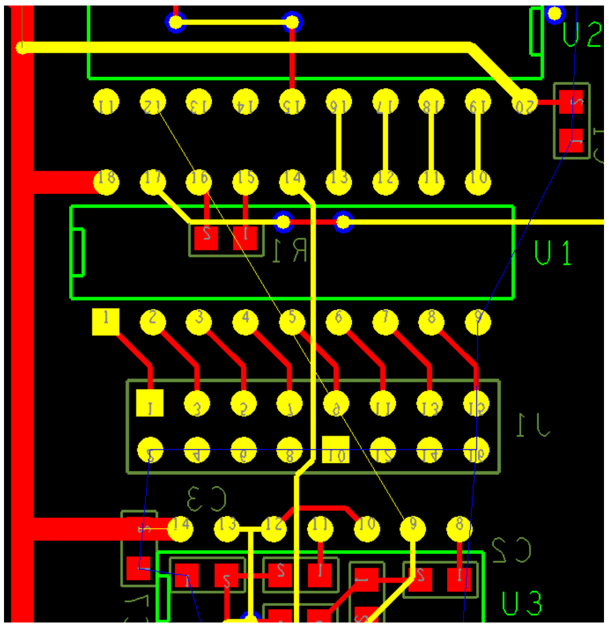


- **Move.** Clines only. This allows you to move the entire connected line.
- **Copy.** Clines only. Simply makes a copy of the entire connected line that you can then place elsewhere. The copied track doesn't retain the parent netname, so if you have a number of identical traces, rather than route each one, just route one of them, copy it and move it to the next position. For example, this could be done with the tracks on U1-2 to 8 (try it).
- **Spin.** Rotates about the mouse point, just like you've already done with several footprints.
- **Change Width.** Segments and Clines. Accesses a simple dialog to specify a new track width. This is used frequently.
- **Mark as Fanout.** Clines only.
- **Fix.** Clines only. This locks the trace in place, so it can't be moved or modified. We seldom use this when manual routing, but when this property is set, autorouters can't change it. Once set, this option changes to *Unfix* when the trace is again highlighted (try it).
- **Property edit.** Clines only. Brings up a dialog allowing you to change specific properties attached to the trace.
- **Show element.** Segments and Clines. A very useful option that should be used when you want instant information about a segment or track (try it). This option is available with most other filtered object too.

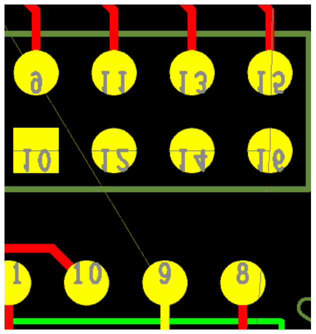
So let's get some mileage from this first net! Rip it and route it from the J1-1 end instead. Then try selecting either segment and changing the track width for just that segment. Rip it again, route the track, and then try changing the width for the entire connection. After routing the track, try adding a third segment. Similarly experiment with the show element, fix and copy commands. Finally, rip it, change to the bottom layer and route it as should be based on the views provided inAppendix-C**.** Change back to the top layer and continue to route all the top layer nets as shown in the pdf file. After this, change the active layer to the bottom, and route all nets as shown in the handout. Note particularly the several components placed beneath U3. Don't worry about the ground nets since we will rely on a copper pour on the bottom layer to connect these. Make the +4.5V nets on both layers 50 mils. Don't route the few nets requiring vias as we'll do that next.

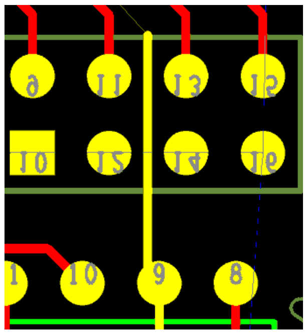
Routing traces on Two layers

We often can't successfully route a net on only one layer, so we have to route it "via" another pin that allows us to go under or over obstructing tracks. An example is shown for the track between U2-12 and U3-9 that also connects to U3-4. We will route this net now using a technique called **"tacking"** and employ two default 13mil vias, as shown in the tutorial handout. Later we'll change these to 20 mils, since the board house we'll be sending this PCB to won't drill anything less than 20 mils (note: this is the actual drill size used).

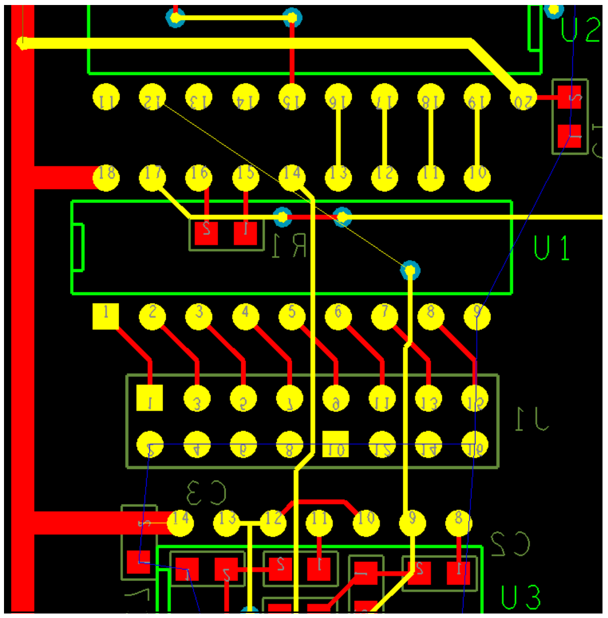


The screen shot left shows the net we want to route going from U3-9 up to U2-12 near the top. Note I've meddled with the pin, ratsnest and net colors to make things more cleary identifiable - just for this example[6]. Route and place a first vertex above and between pins 12 and 14 up through 11 and 13 on the pin header, J1. Zoom in to precisely position the partial segment precisely between these pins. This is shown below in the two side-by-side sequence screen shots.

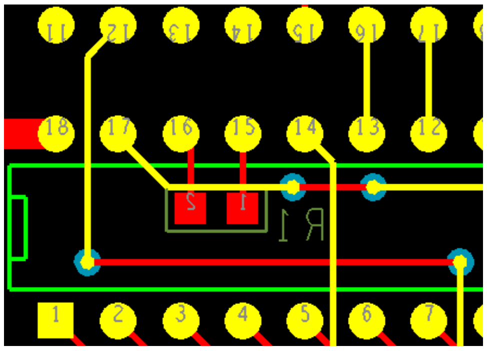




Continue routing on up through pins U1-7 and 8 (zoom out as necessary). Complete the track just beyond pins 7 and 8 leaving sufficient clearance for a via (Allegro won't let you add a via that violates DRC clearances – 8mils in this case). Add the wanted via at this ending vertex by simply positioning the the mouse on it and double-clicking. Allegro automatically place a via at this point and change to the bottom layer, since that's what Allegro thinks you want to do. Note there is no sub-context for this action, *i.e.* it isn't necessary to right-click and select *Done*. The partial route with the new via will now look like the picture below left (except the active layer will be the bottom).



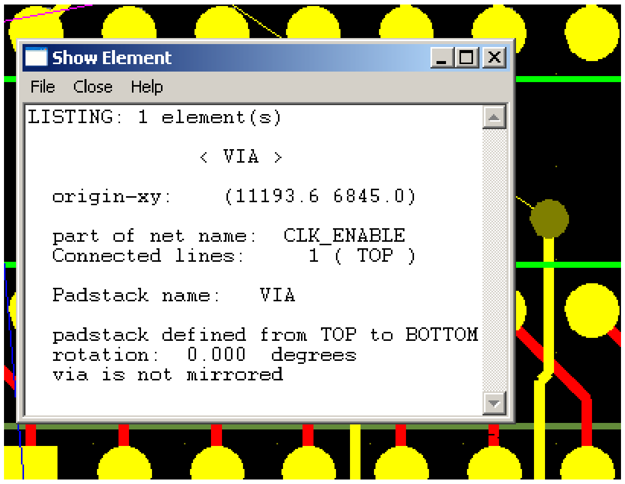
MANUAL TACKING - Route the last two segments using a technique called *tacking*. Rather than placing a vertex first, immediately double-click the mouse instead. Allegro will respond by placing a vertex *and* a via *and* immediately change the active layer to the bottom so you can continue with the next segment. Do this now and complete the track all the way to U2-12, shown below.



[6] +4.5V net only *Net* and *Rats* are colored; Gnd net excludes *Pins*; *Rat to-bottom* is bright yellow (this highlights the one we're looking at here from U3-9 to U2-12; *Package_Geometry / Pin_Number* is light gray.

Now rip this entire track and try routing it again, but using only tacking all the way. This manual technique, placing vias and changing routing layers on the fly, is aptly called "tacking", similar to the way a sailboat is piloted when sailing into the wind. You will probably have to go back and finely position some segments (to get them between the DIP pins) and one or both of the two vias. Segments are moved and vertex's added using the *Cline Segs* filter. Vias are moved by simply setting the *Vias* filter; setting both of these is possible, but some is skill is needed to avoid selecting the wrong object. First learn how to select and move vias using only the vias filter, then try enabling both of them and experimenting with selecting and moving vias and segments alternately.

Once nets are connected to vias they are treated as electrical objects. Select either of the two vias you just routed and select *Show Element*. I've done just that with the first via; the message is shown below.

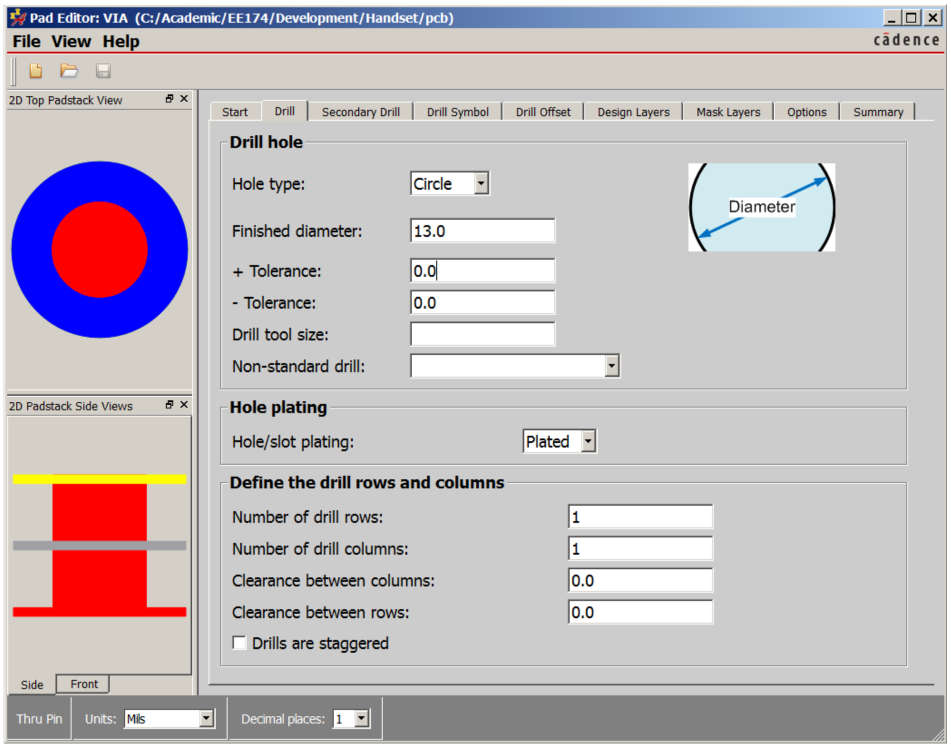


```
LISTING: 1 element(s)

            < VIA >

  origin-xy:    (11193.6 6845.0)

  part of net name:  CLK_ENABLE
  Connected lines:     1 ( TOP )

  Padstack name:   VIA

  padstack defined from TOP to BOTTOM
  rotation:   0.000  degrees
  via is not mirrored
```

Notice the message states this via is "part of net name: CLK_ENABLE", making it an electrical object. Besides vias, get in the habit of using the *Show Element* option to gain a good idea of how Allegro treats and uses any object, like segments, footprints etc.

Padstack Designer

The vias you just placed are most likely 13mil padstacks. The board house we want to eventually send this PCB out to won't drill padstacks less than 20mils, so we will need to change these vias. I'll show you the relationship between vias and padstacks now. Select one of the two vias and look at its' corresponding padstack in the *Padstack Designer*. Do this by first highlighting or selecting the via (set the filter to "vias" only) and choosing **Modify design padstack / Single instance** from the right-click context menu. This will bring up the padstack editor dialog shown . This is used to create new padstacks and modify existing ones. Here, we're using it to verify a particular padstack being currently being used as a routing via. I will discuss the general topic of designing padstacks in more detail in the next laboratory, including the topic of thermal reliefs. For now, just know how to access this dialog to quickly inspect particular padstack parameters. Later in this lab, we'll use it to change several drill sizes and adjust annular ring sizes.



Drill hole

Hole type:	Circle
Finished diameter:	13.0
+ Tolerance:	0.0
- Tolerance:	0.0
Drill tool size:	
Non-standard drill:	

Hole plating

Hole/slot plating:	Plated

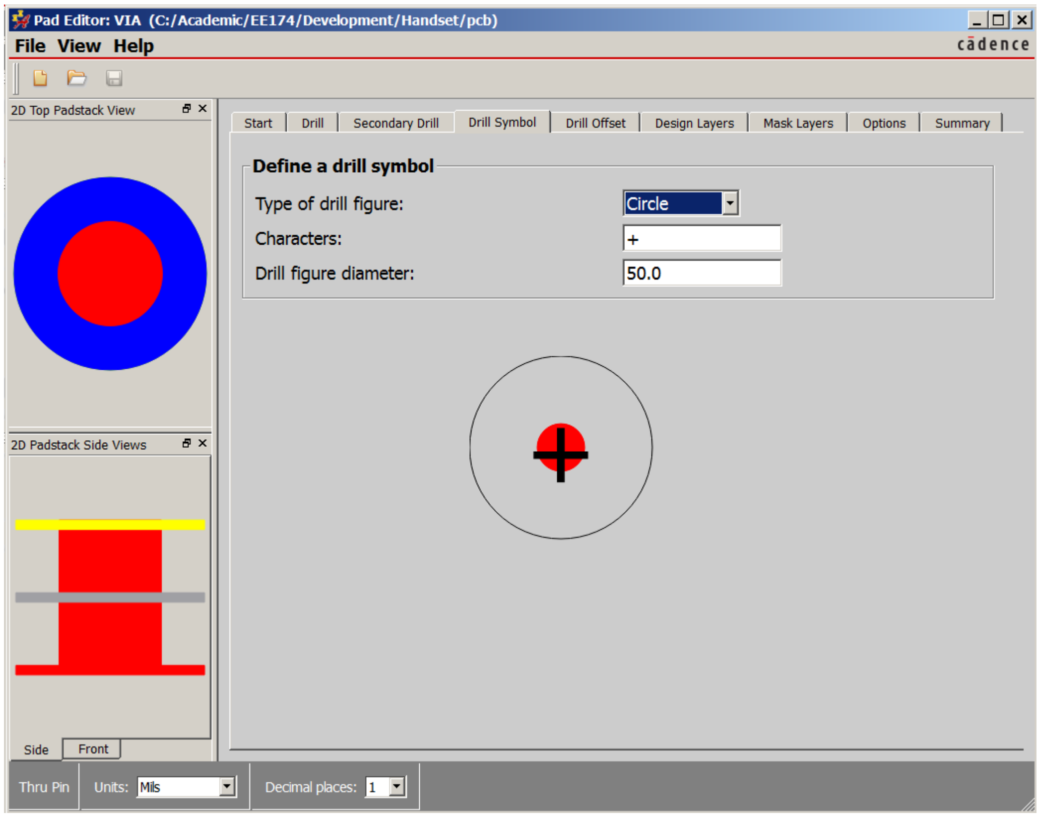
Define the drill rows and columns

Number of drill rows:	1
Number of drill columns:	1
Clearance between columns:	0.0
Clearance between rows:	0.0

☐ Drills are staggered

Units: Mils Decimal places: 1

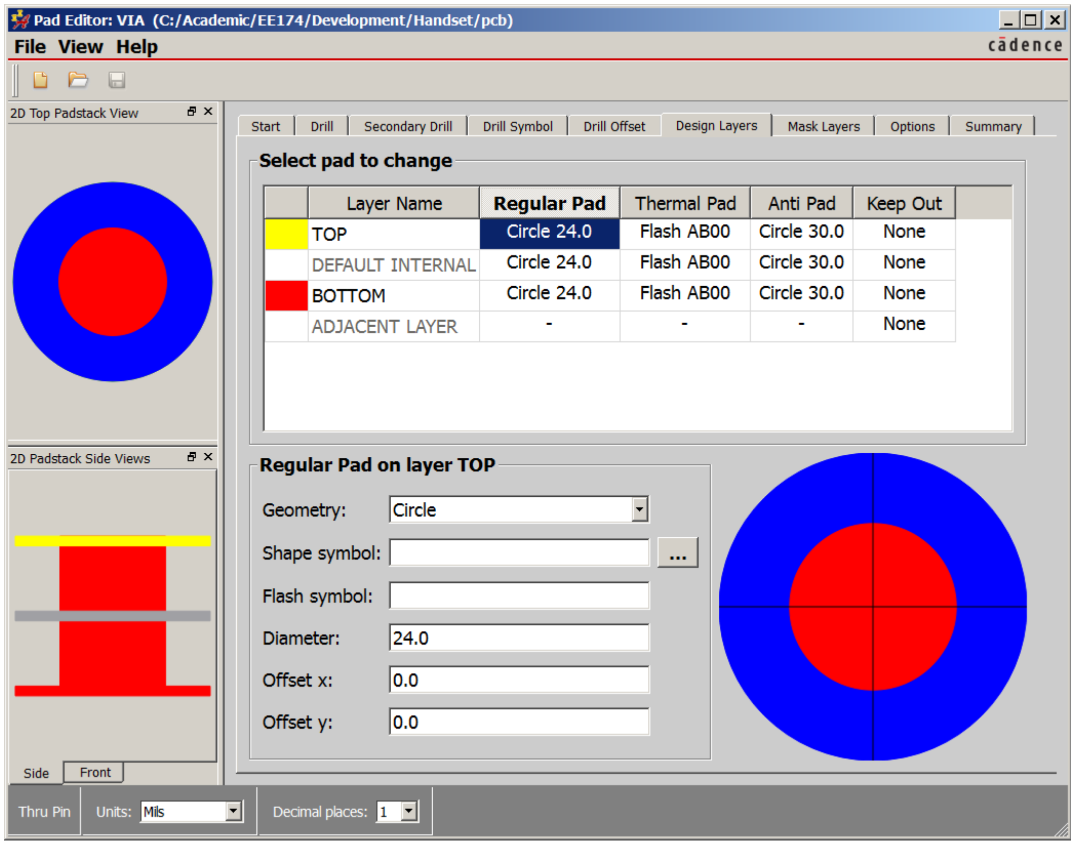
From the *Drill* tab, we see in the *Drill Hole* and *Hole Plating* blocks this via is a plated through 13.0 mil circular drill hole providing an electrical connection through the PCB. The *2D Top Padstack View* block displays the padstack's 13mil drill as a red pad with the larger blue pad it drills through. These two overlaid pads are proportionally correct. The blue region beyond the red pad is the actual relative amount of copper that appears around the drilled hole. This "donut" is called an *annular ring* and must be at least 5 mils to handle off-center drilling variations. The *2D Padstack Side Views* below it hows this to be a via that drills completely through the PCB. The thin gray region in the middle of the cross-section is a short hand graphic for any possible inner copper layers that may also be present; in this 2-layer design there are none.



Define a drill symbol
Type of drill figure: Circle
Characters: +
Drill figure diameter: 50.0
Units: Mils
Decimal places: 1

Select the *Drill Symbol* tab to see the "Define a drill symbol" dialog as shown. This specifies how the padstack is visually represented on one of the special documentation layers, namely as a 50mil circle with a "+" character inside. You will understand how this is used during the post-processing phase at the end of the lab. For now I'll show you how the rest of the padstack is specified.

Select the *Design Layers* tab to view how Allegro specifies the actual etch layer (copper) pads, Top and Bottom.


Pad Editor: VIA (C:/Academic/EE174/Development/Handset/pcb)

Select pad to change

	Layer Name	Regular Pad	Thermal Pad	Anti Pad	Keep Out
🟨	TOP	Circle 24.0	Flash AB00	Circle 30.0	None
	DEFAULT INTERNAL	Circle 24.0	Flash AB00	Circle 30.0	None
🟥	BOTTOM	Circle 24.0	Flash AB00	Circle 30.0	None
	ADJACENT LAYER	-	-	-	None

Regular Pad on layer TOP

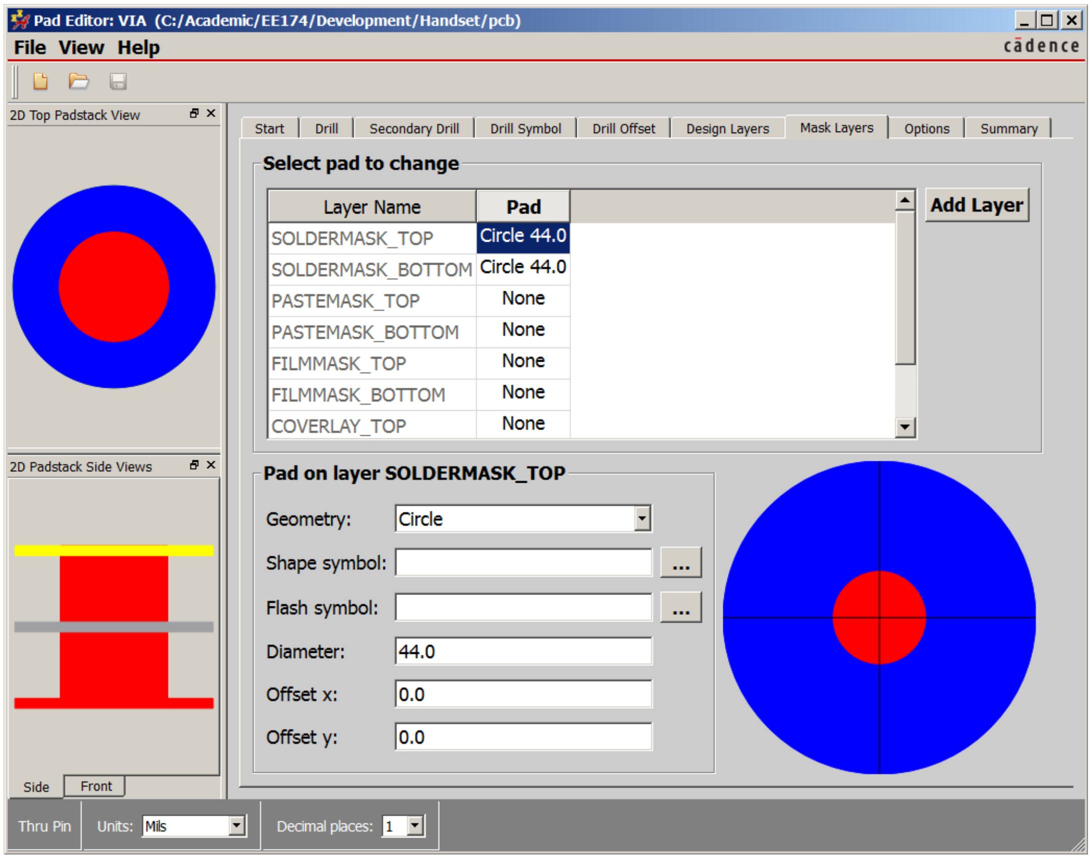
Geometry:	Circle
Shape symbol:	
Flash symbol:	
Diameter:	24.0
Offset x:	0.0
Offset y:	0.0

Thru Pin Units: Mils Decimal places: 1

The tab reveals that both pads, top and bottom sides are 24 mils wide; this is shown in the *Regular Pad* column. This means the annular ring must be 5.5 mils thick, shown as the yellow (top) and red (bottom) regions in the *Padstack Side View* cross section view shown. When a hole is first drilled, it is exactly that of the drill, 13 mils. Plating copper into it and contacting the connecting pads, causes the effective diameter to shrink by an amount known as the *plating allowance*. Typically this inner plating is about 1.5 mils thick, causing the original drilled hole to shrink by 3 mils. Another way of looking at this is that the pad's annular ring increases by 3 mils to 8.5 mils after plating.

Selecting each of the three *Regular Pad* fields (mouse click) causes the lower block to display details associated with that particular layer. Above, I have selected the "Circle 24.0" field on the top layer. Below, the detailed red region depicts the drilled PTH hole with its resulting blue annular ring. *Anti Pad* refers to an "empty" pad overlay that defines a safety region, an annular ring about the pad which contain no copper. Since the anti pads are 30 mils and the regular pads are 24, this annular ring is 3 mils thick.

Now select the *Mask Layers* tab. These define the remaining non-etch or non-conductive layers that will be discussed in lab 2.1.



Select pad to change

Layer Name	Pad
SOLDERMASK_TOP	Circle 44.0
SOLDERMASK_BOTTOM	Circle 44.0
PASTEMASK_TOP	None
PASTEMASK_BOTTOM	None
FILMMASK_TOP	None
FILMMASK_BOTTOM	None
COVERLAY_TOP	None

Pad on layer SOLDERMASK_TOP

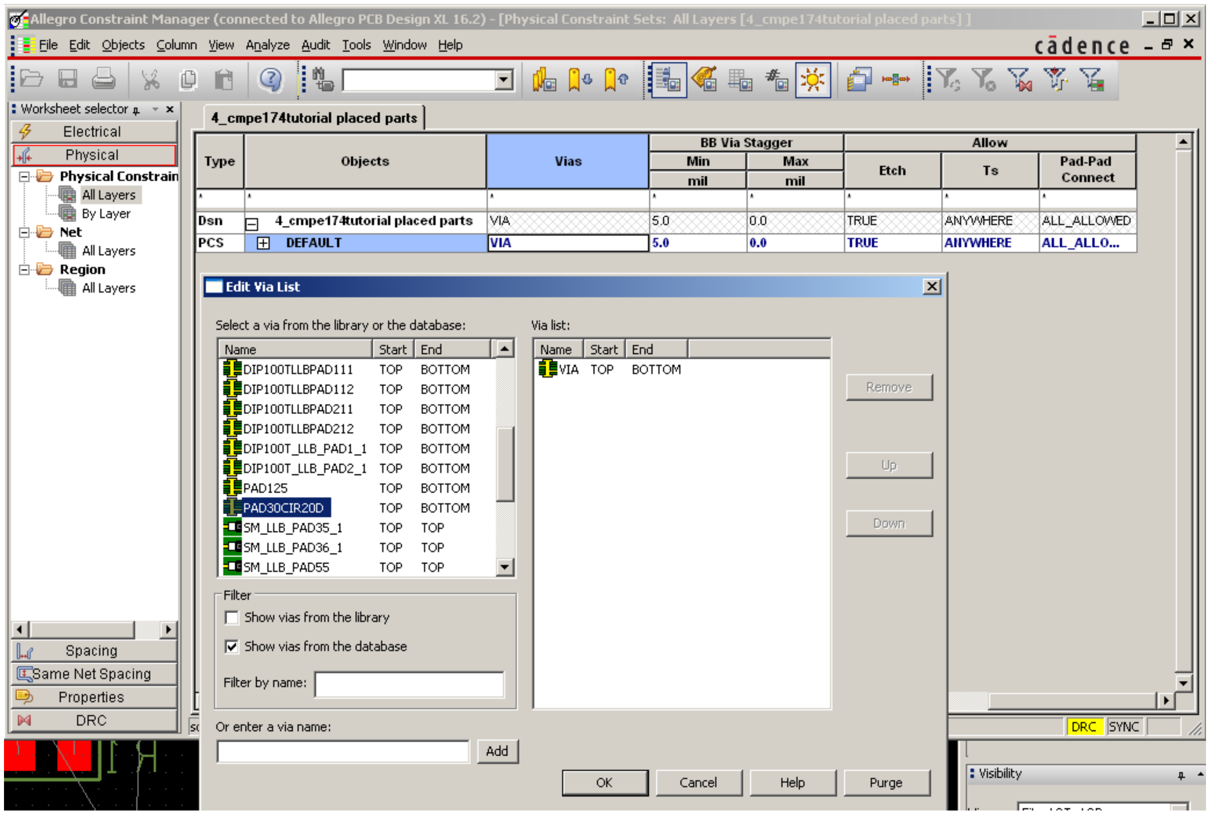
Geometry:	Circle
Shape symbol:	
Flash symbol:	
Diameter:	44.0
Offset x:	0.0
Offset y:	0.0

Units: Mils
Decimal places: 1

Summarizing our observations about this via, we find it to be a 13 mil via with 5.5 mil annular rings *before plating* and 8.5 mil *after plating*. Its pad size is 30mils and with a 44mil soldermask on the top and bottom layers. Note the soldermask is what its name implies: a layer of enamel paint preventing the flow of solder. Hence this via has an unprotected area 6 mils beyond the 30mil copper pad.

Changing the Default Via

Allegro provides many ways to work with vias. I'll dicuss several of them that are most pertinent to our needs: 1) changing the default via. 2) changing vias individually or in groups. In the next lab, I'll show you how to make a via from scratch. First, let's change the default via. This is done by setting a global entry in the physical section of the *Constraint Manager*; it is accessed either by menu sequence **Setup ⟶ Constraints ⟶ Physical…** or by using the icon and selecting the *Physical Constraint Set* in the Worksheet Selector on the left-side of the form. Once the dialog appears, select sub-option *All Layers* and scroll sideways to find the **Vias** column. Select the entry in the Vias column coinciding with the **Default** row; the *Edit Via List* dialog will appear. The Via List may have more than one entry in it, but the topmost line will be *Via*. Remove all entries in this list and add the *PAD30CIR20D* padstack from the list in the left column. Double-clicking it will move it over to the Via List. Select this from the "database" rather than the "library". Remember, "database" is Cadence-speak for "your current design". The default Allegro library is quite long, and this particular via is a 20 mil drilled padstack with an annular ring size of 8 mils after plating. I copied it to your symbols folder. Some clarifying screen shots that will help you are shown following.


Allegro Constraint Manager (connected to Allegro PCB Design XL 16.2) - [Physical Constraint Sets: All Layers [4_cmpe174tutorial placed parts]]
File Edit Objects Column View Analyze Audit Tools Window Help
cādence
Worksheet selector
Electrical
Physical
Physical Constrain
All Layers
By Layer
Net
All Layers
Region
All Layers
Spacing
Same Net Spacing
Properties
DRC
4_cmpe174tutorial placed parts
DRC
SYNC
Visibility

Type	Objects	Vias	BB Via Stagger Min mil	BB Via Stagger Max mil	Allow Etch	Allow Ts	Allow Pad-Pad Connect
*	*	*	*	*	*	*	*
Dsn	4_cmpe174tutorial placed parts	VIA	5.0	0.0	TRUE	ANYWHERE	ALL_ALLOWED
PCS	DEFAULT	VIA	5.0	0.0	TRUE	ANYWHERE	ALL_ALLO...

Edit Via List

Select a via from the library or the database:

Name	Start	End
DIP100TLLBPAD111	TOP	BOTTOM
DIP100TLLBPAD112	TOP	BOTTOM
DIP100TLLBPAD211	TOP	BOTTOM
DIP100TLLBPAD212	TOP	BOTTOM
DIP100T_LLB_PAD1_1	TOP	BOTTOM
DIP100T_LLB_PAD2_1	TOP	BOTTOM
PAD125	TOP	BOTTOM
PAD30CIR20D	TOP	BOTTOM
SM_LLB_PAD35_1	TOP	TOP
SM_LLB_PAD36_1	TOP	TOP
SM_LLB_PAD55	TOP	TOP

Via list:

Name	Start	End
VIA	TOP	BOTTOM

Filter

[ ] Show vias from the library

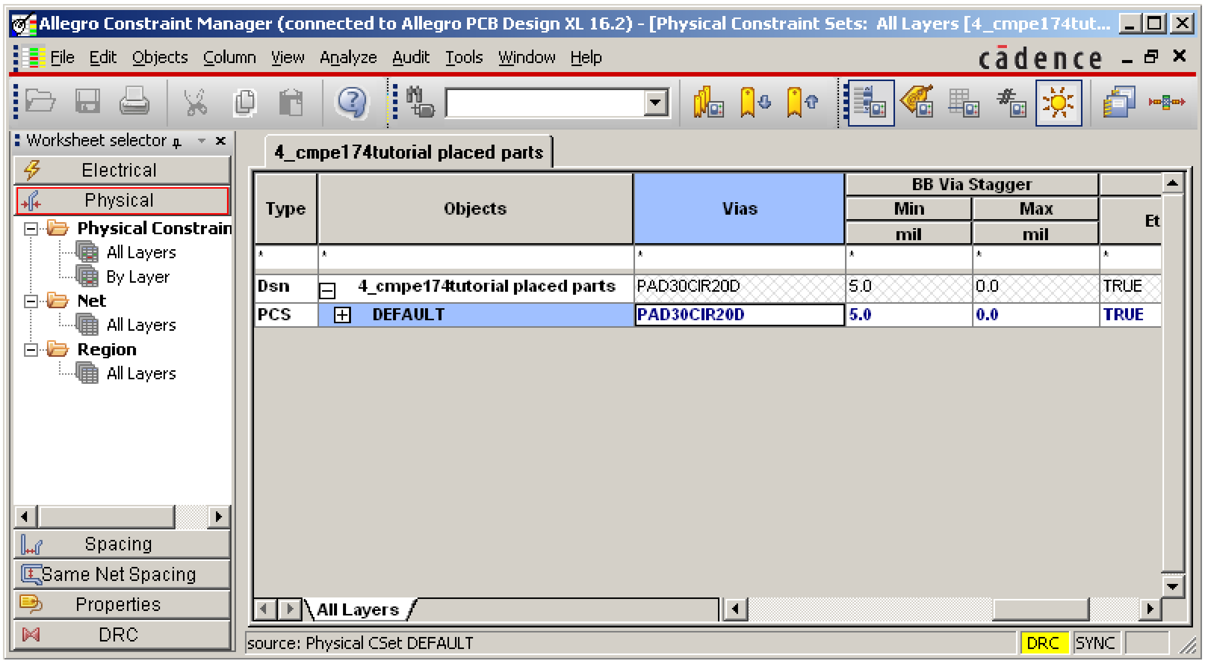
[✓] Show vias from the database

Filter by name:

Or enter a via name:

The shot above shows things before I removed the *Via* from the Via List and replaced it with the 20 mil padstack. Note the name "Via" refers to a particular padstack having that name. It shows up at the bottom of the available selections, whether we select the library , database or both. Notice the blue entry, **VIA,** where the Default row and Vias column intersect; that's the one to click (select).

The following shot shows the result after pressing OK and updating the field. It should now read **PAD30CIR20D**. Notice this nicely encodes the padstack: 20D for 20 mil drill; 30CIR for 30 mil circular pads. Mentally this results in a 5 mil annular ring before plating and, with a standard 3 mil plating allowance, 8 mils after plating.


Allegro Constraint Manager (connected to Allegro PCB Design XL 16.2) - [Physical Constraint Sets: All Layers [4_cmpe174tut...
File Edit Objects Column View Analyze Audit Tools Window Help
cādence
Worksheet selector
Electrical
Physical
Physical Constrain
All Layers
By Layer
Net
All Layers
Region
All Layers
Spacing
Same Net Spacing
Properties
DRC
4_cmpe174tutorial placed parts
All Layers
DRC
SYNC

Type	Objects	Vias	BB Via Stagger Min mil	BB Via Stagger Max mil	Et
*	*	*	*	*	*
Dsn	4_cmpe174tutorial placed parts	PAD30CIR20D	5.0	0.0	TRUE
PCS	DEFAULT	PAD30CIR20D	5.0	0.0	TRUE

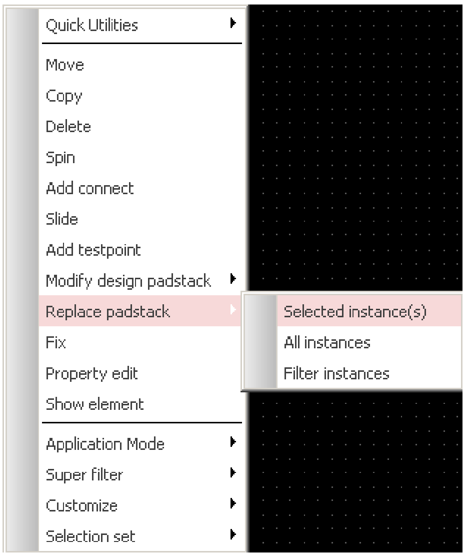
source: Physical CSet DEFAULT

Rip the entire track you did earlier and try routing it again; the vias placed should be 20 mil padstacks. I'll now show the second way to edit vias, individually or in groups. Begin by yet again ripping the entire track and resetting the default via back to *Via* in the Physical Constraints worksheet. Proceed to route the track again; it should have 13 mil vias as before (you should be getting pretty good at this by now!).

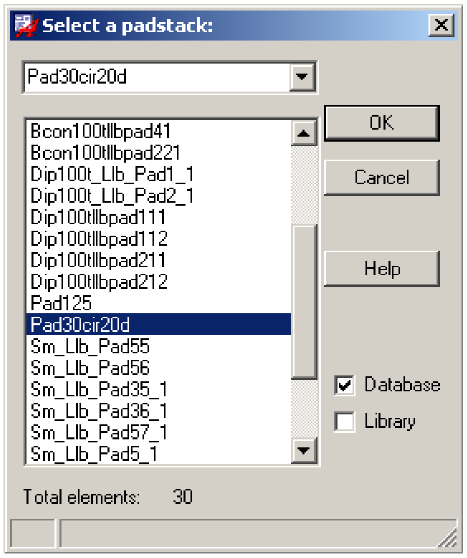
Changing Vias by Instance and Groups

Allegro gives us two ways to access the editing environment to change placed vias:

1. Use the context sensitive menu, but… *you must be in General Edit mode* to see "Replace Padstack" option



Choosing *Selected instance(s)* will replace only the selected via, while *All instances* will result in every occurance of the selected via in your design being replaced. Shown following is what the resulting *Select a padstack:* dialog looks like for either case. Here, I limited the list to the current design and selected the wanted 20 mil replacement padstack. Pressing OK will finish the operation.



Pad30cir20d

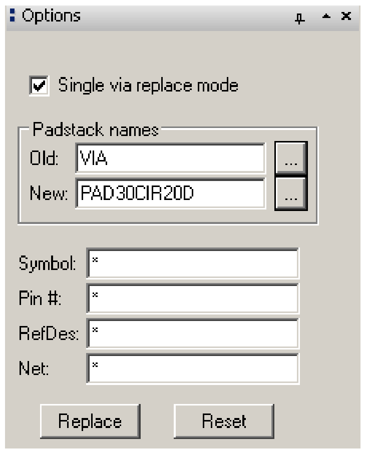
Bcon100tllbpad41
Bcon100tllbpad221
Dip100t_Llb_Pad1_1
Dip100t_Llb_Pad2_1
Dip100tllbpad111
Dip100tllbpad112
Dip100tllbpad211
Dip100tllbpad212
Pad125
Pad30cir20d
Sm_Llb_Pad55
Sm_Llb_Pad56
Sm_Llb_Pad35_1
Sm_Llb_Pad36_1
Sm_Llb_Pad57_1
Sm_Llb_Pad5_1

[✓] Database
[ ] Library

Total elements: 30

Note that Menu sequence **Tools ⟶ Padstack → Modify Library Padstack…** also accesses the same dialog except the choice is locked to only “Library”. This is available in modes other then General Edit.

2. Main menu sequence **Tools ⟶ Padstack → Replace…** causes a versatile persistent dialog to appear in the Options panel. Here, vias in groups or singly (or equivalently, padstacks) can be changed easily and quickly.

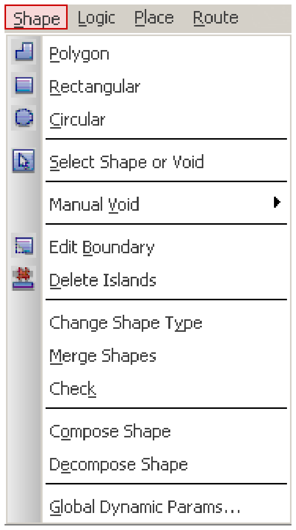


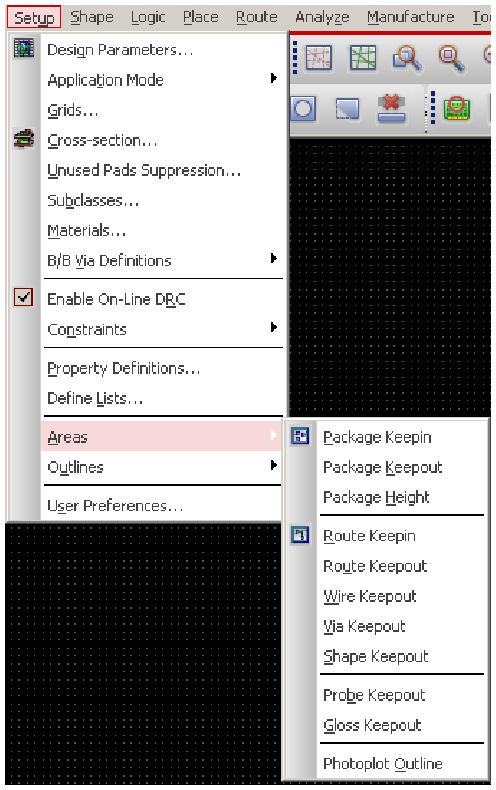
Selecting any padstack or via will pre-load the *Old:* field, or it can be selected by pressing the button; this will bring up the *Select a padstack:* dialog above. Select the replacement for the *New:* padstack the same way (using the dialog). If you don’t check the *Single via replace mode* checkbox, pressing the *Replace* button will change all instances of the old padstack to the new in your design; otherwise, only the single instance will be replaced.

After being sure your default via has been changed to 20 mils, go ahead and finish routing all of the nets except the GND net. We’ll do that next.

4.7 Working With Shapes

So far we’ve routed connections on etch layers using “connect lines”, but often we need to create areas of copper that aren’t traces or pads, like rectangles, squares, circles, polygons etc. Allegro has an interesting set of objects to address this challenge; they’re called *shapes*. Shapes are geometric planar objects that may or may not have electrical properties attached to them. For example, we can use them to create copper areas tied to specific nets, or no net all. Or we might use one to define an area whose purpose is just the opposite: to keep copper *out*; these are called *keepouts* and they define regions called *voids* (this colorful vacuous term applies to any region on a PCB where copper is deliberately excluded). All of these, and many more, are generically called “shapes”. They are handled with a versatile software toolset accessed under the **Shape** and **Setup→Areas** pull-down menu options. Take a moment to look at these now:





As usual, the icons noted flag their presence on several of the Toolbars as well.

We will create three shapes for use in our current PCB design, two different kinds of copper areas, and one specific kind of keepout. Indeed, notice the *many* kinds of keepouts! Most of these are meant for use with Allegro’s powerful autorouter. We will use one of them: *Shape Keepout*. Generally, shapes are pretty easy to work with, the key is know which to use for a particular purpose, and where to find it in the menu labyrinth.

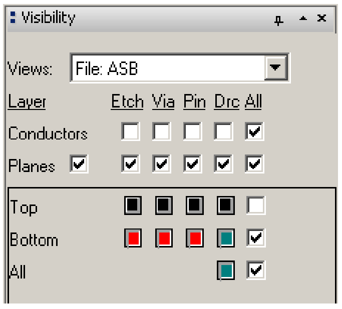
Copper Pours

The GND net was purposely left unrouted, since we want to use a plane of copper on the bottom layer to connect all of the padstacks, pads and vias, tied to it. We call these planar areas *copper pours*, a generic industry term. Cadence calls them “dynamic copper shapes”. This kind of shape can only be added to Etch layers.

Understanding how Allegro makes them requires some basic knowledge of the later manufacturing process. They are created much like single, connect-line, circuit traces are: a light-pen, having a circular beam of UV light equal to the trace width, called the *aperture*, is moved on an X-Y photoplotter to expose light sensitive material, called *photoresist* (as an aside, did you notice that partially routed nets always had a dangling end that was round? Now you know why; that round end was equal to the aperture diameter). After the exposure step, the photoresist layer is then chemically “developed”: exposed areas remain and everything else dissolves away. The exposed areas protect the underlying copper from being removed in the next step, acid immersion. Finally the solid photoresist is removed leaving only the wanted PCB interconnecting traces. By successively laying down ordered, constant width horizontal traces, and overlapping them by half the trace width, areas of solid copper can be geometrically defined. This technique is called *raster scanning*. Each light pen, having a fixed diameter aperture to be moved about by an XY photoplotter, is called a *draw*. How finely detailed the perimeter (the “shape”) can be resolved clearly depends on the light aperture; finer detailing requires narrower trace widths. As you will shortly see, Allegro defaults to a trace width sufficient for 4 mil details for pours (typically about 8 mil draws). Making it smaller than this typically isn’t necessary since the added resolution is seldom needed and just makes screen redraw times longer, and the associated output Gerber files bigger; don’t’ be tempted to decrease this just to “make things look better” – you probably can’t see the difference anyway.

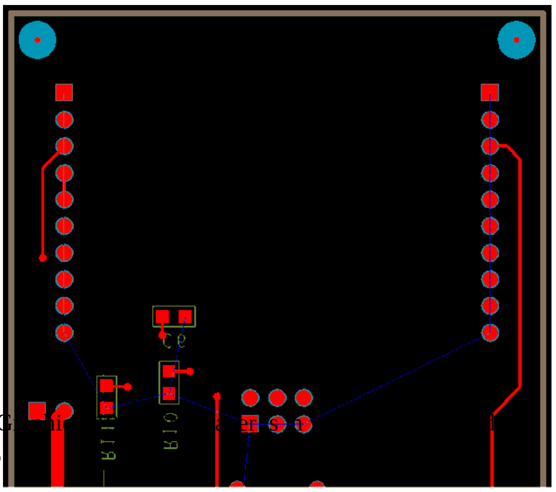
One last point needs to be made. I mentioned that traces and raster scans are created using *draws*, basically by moving light pens. By constrast, pads of all geometries (mostly squares, rectangles and circles) are created by holding a single light pen, with the wanted geometry, fixed at one position and “flashing” the photoresist stroboscopically. These so-called *flashes* complement *draws*. All photoresist exposures are done using one of these two basic types[7]. Most shapes are implemented with raster scan draws, but some, like thermal-reliefs are done with flashes.

Refer to the color handout showing the red copper pour as a polygon on the bottom layer. At the top of the board, we’ll first create a simple rectangular area, make some observations about it, and then proceed to let you create the remaining areas. Begin by setting the ASB view and rapidly turning off everything on the top layer by using the checkbox options in the Visibility panel. Simply uncheck it in the *All* column as shown below. These Visibility options are really fast and easy to use and learn. Notice too they only affect *Etch* layers – those having tangible copper. Changing visibility of any other sub-class must be done either in Color Dialog or the Options’ panel. Take a moment to experiment with the checkboxes to get an experimental, *visual*, appreciation for their effects, and use them anytime you like. Remember, earlier, the point was made that only the Etch, Via and Pin classes collectively control actual PCB copper. Note that *Planes* controls inner layers, which we don’t have, so the checkboxes do nothing (try them). If you want, just uncheck the left-most “Planes” checkbox and the five useless options will disappear.



Views: File: ASB

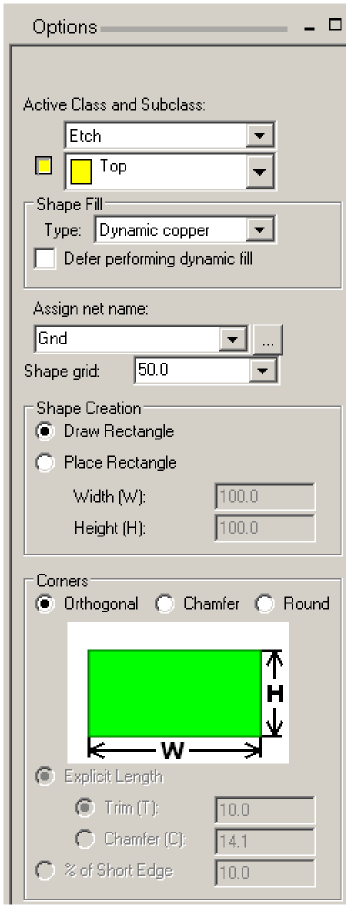
Layer		Etch	Via	Pin	Drc	All
Conductors		☐	☐	☐	☐	✓
Planes	✓	✓	✓	✓	✓	✓
Top		■	■	■	■	☐
Bottom		🟥	🟥	🟥	🟦	✓
All					🟦	✓



[7] See Gerber Rs-274X Format, User’s Guide, Barco G

©

I’ve shown a screen shot of what your desktop should look like (above right). We’ll place the first shape on the bottom layer in this area. Select the *Rectangle* tool from the shape toolbar (or menu sequence **Shape→Rectangle)**. Focus on the changes in the Options’ panel, as shown:


Options

Active Class and Subclass:
Etch
Top

Shape Fill
Type: Dynamic copper
☐ Defer performing dynamic fill

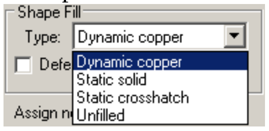
Assign net name: Gnd
Shape grid: 50.0

Shape Creation
[✓ Draw Rectangle]
Place Rectangle
Width (W): 100.0
Height (H): 100.0

Corners
[✓ Orthogonal] Chamfer Round

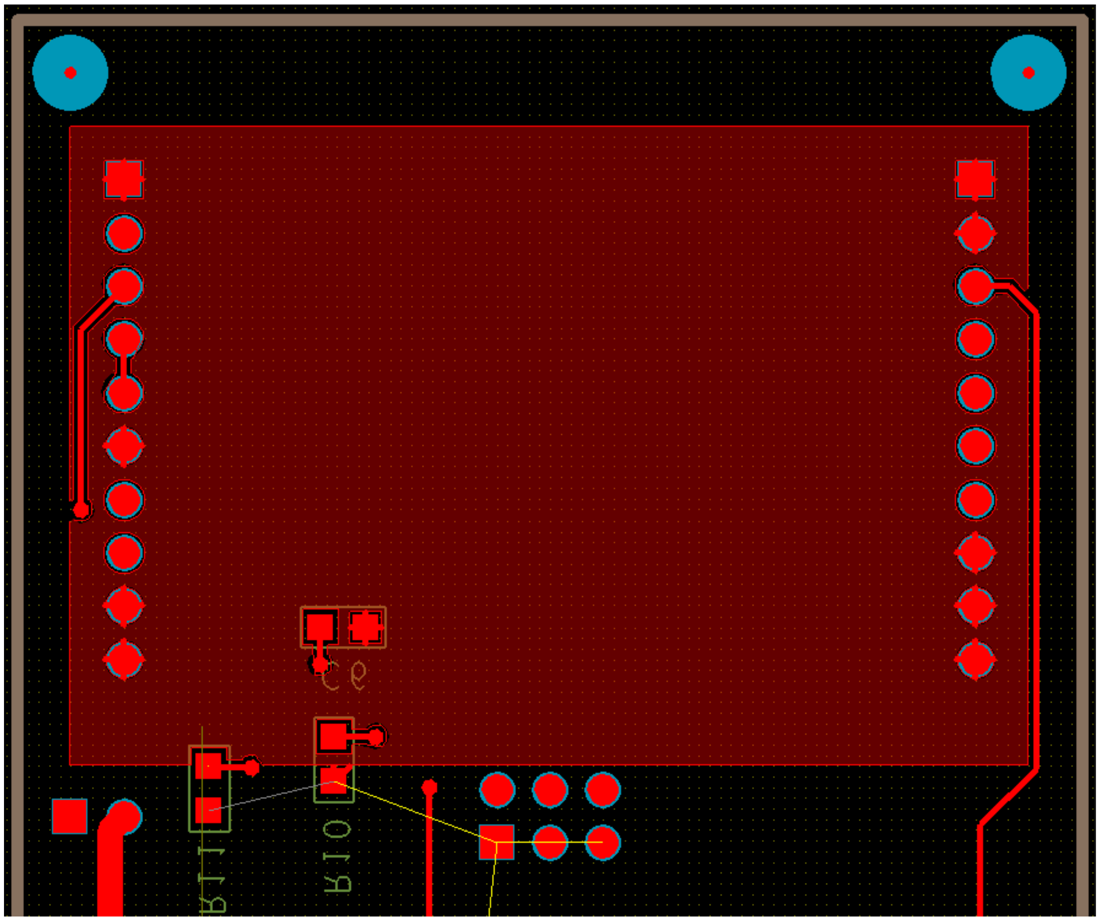
[✓ Explicit Length]
[✓ Trim (T):] 10.0
Chamfer (C): 14.1
% of Short Edge 10.0

Set the active layer for the bottom copper and select the Shape Fill *Type* as “Dynamic copper”. Look for a moment at the “fill” choices in this listbox, shown below, where we’re interested in understanding the difference between the first two options:

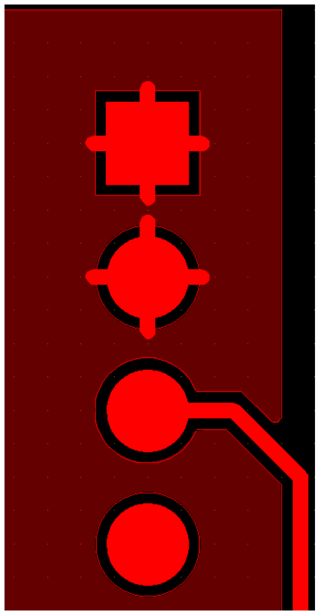

Shape Fill
Type: Dynamic copper
Dynamic copper
Static solid
Static crosshatch
Unfilled

Succinctly, dynamic shapes *autovoid*. Static shapes don’t, unless we *manually void* them. Everytime you make a change on the board, Allegro responds automatically by *re-pouring* any relevant dynamic copper shapes to accommodate the physical change. You’ll see the great benefit of this shortly.

Since the dyanamic shape we want to place now will be an electrical object, assign it to net **GND** shown (the button will bring up a dialog with all nets in the current design; choose the GND net). The last option, *Shape grid*, allows us to temporarily set the Etch grid only while placing this shape. The current setting is usually way too fine anyway and must otherwise be changed on the *Define Grid* dialog – an annoying inconvenience. Allegro allows up to five temporary values to be directly entered in the pull-down list (they persist until the program is closed). Enter 50 as shown and it will be added to the two default entries: *Current grid* (this is the current Etch grid), and *none.* The latter is “gridless”, where the shape doesn’t snap to the grid at all; think of this as “free-form” drawing. Now draw the rectangle. It should come out looking like the screen-shot following (be sure to right-click and select Done):

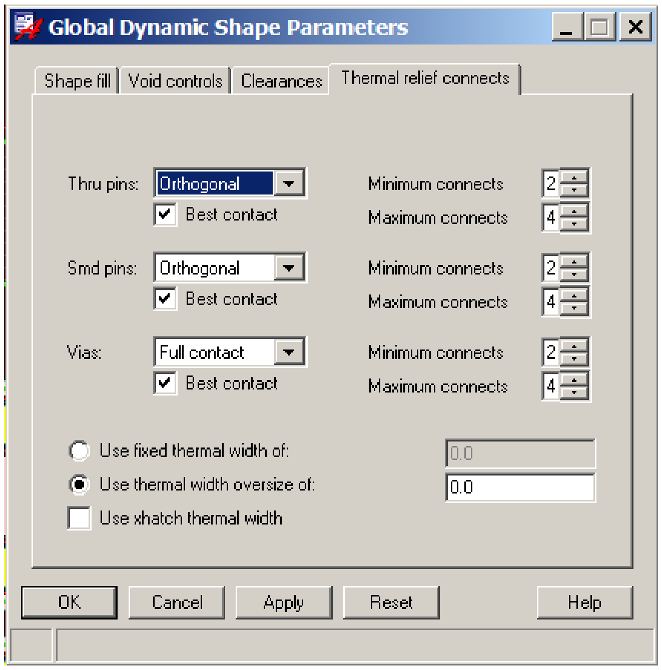


Notice that all pins on the two single-row headers are now within the pour's area. The following closeup of the upper-right region clearly shows what happened to every pin. To show *only* the shape and padstacks, I set the view to "Film: Bottom" in the Visibility panel; that turned off the possibly confusing blue soldermask layer.



Thermal Reliefs and Voids. Carefully observe the voids (clearances) around the trace and four pads shown. Note especially the two upper pads, where they have been automatically connected to the new copper plane. These are called *thermal reliefs* (also known as *solder thieves* or *solder reliefs*). Their purpose is to limit heat transfer while maintaining good electrical connectivity. They are usually seen as four very short and narrow traces, like what you see here, arranged symmetrically like spokes around a wheel, providing good low-impedance electrical connections to some surrounding large metal surface that deliberately limits heat transfer during the soldering process. Circular and square pads associated with pins are a good example. The idea is to create 4 narrow arcs at 0, 90, 180 and 270 degrees around the pad inter-connecting the padstack, thereby making it possible to heat it and solder an associated *through-hole* part. PCB's designed for automatic soldering processes *must always* have these solder reliefs, but we sometimes omit them for specific purposes. Allegro places solder reliefs automatically. Deliberately omitting them is called *flooding* a pin or via. Vias used only for routing don't need thermal reliefs, since there is *nothing to solder them to*. Therefore, padstacks created for use solely as routing vias shouldn't have thermal reliefs. On the next page, I'll show you where these configuration settings can be found.

Thermal relief settings are defined on the *Thermal relief connects* panel, as shown following on the Global Dynamic Shape Parameter dialog:


Global Dynamic Shape Parameters

Thru pins: Orthogonal
[✓] Best contact
Minimum connects 2
Maximum connects 4

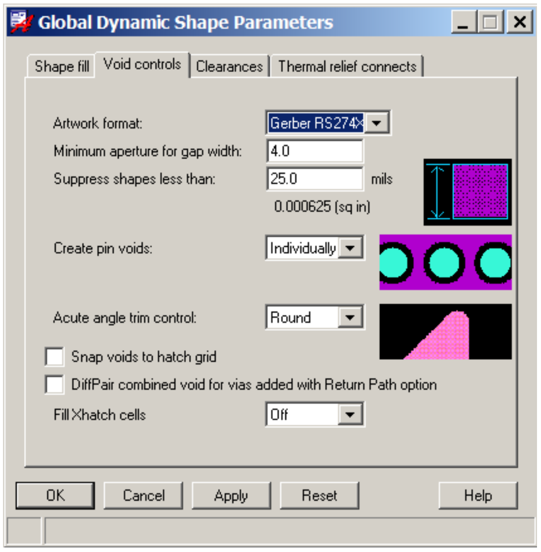
Smd pins: Orthogonal
[✓] Best contact
Minimum connects 2
Maximum connects 4

Vias: Full contact
[✓] Best contact
Minimum connects 2
Maximum connects 4

○ Use fixed thermal width of: 0.0
● Use thermal width oversize of: 0.0
☐ Use xhatch thermal width

This panel is fun to play with, but unless you've got a good reason other than "wow, see what I can do", leave the settings alone. Note that *Vias* have been set for "Full contact". This is equivalent to flooding all routing vias. At this point, you can more clearly perceive Allegro's fine distinction between "Thru pins", surface-mount or "SMD" pins, and "Vias". All of these are considered padstacks. Through pins have connected parts, like sockets or component leads, while vias don't. This means the same library padstack could be used as a pin in one instance and a routing via in another. What we're telling Allegro here is to create thermal reliefs for pins but not for routing vias. SMD pins, also padstacks, define pads on one side of the board only (top or bottom); they have no drilled hole. But since some external part is typically soldered to them, they still need thermal reliefs. Good examples in the present design are the 0805 resistor and capacitor padstacks; these mount small rectangular (80mil by 50mil) components. Note that I will generally conflate through-hole padstacks with vias, and following common industrial practice, frequently refer to Allegro PTH pins and Allegro PTH routing vias as just generic "vias"; they are, after all, both PTH vias. Only their target use varies.

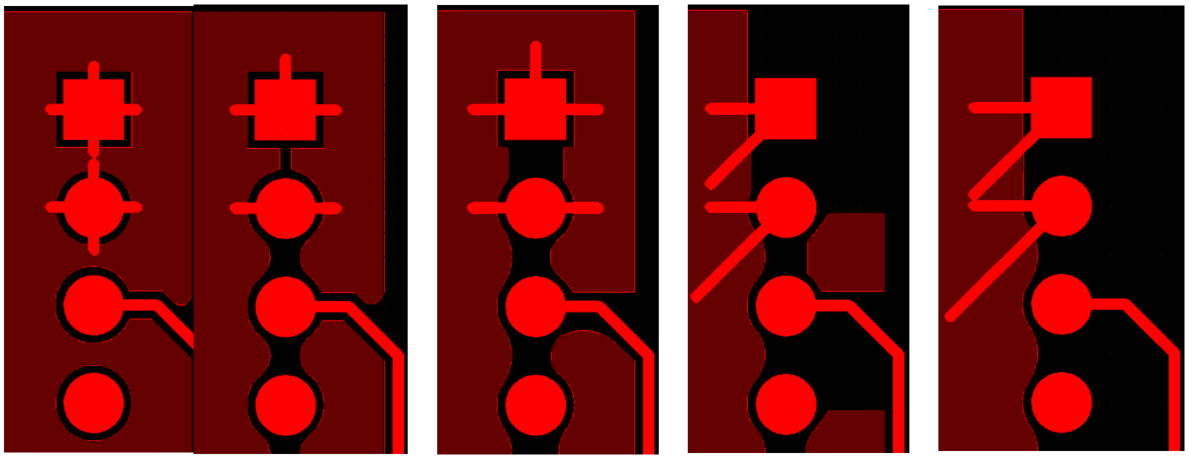
Each of these four example pins have automatic 8 mil clearance *voids* placed around them, as does the single connect-line trace. These voiding widths are specified in the constraints' spreadsheet that were defined in the technology constraint file (*.tcf) we imported at the very beginning based on design complexity level B – 8 mils. Remember too, the relative dimness of the poured area is due to the 39% *Shapes transparency* setting we made earlier in the Color Dialog. You might want to go back and set this to 100% to make the pour appear solid, but be sure to set it back to 39% before proceeding.



Artwork format: Gerber RS274X
Minimum aperture for gap width: 4.0
Suppress shapes less than: 25.0 mils
0.000625 (sq in)
Create pin voids: Individually
Acute angle trim control: Round
☐ Snap voids to hatch grid
☐ DiffPair combined void for vias added with Return Path option
Fill Xhatch cells: Off

Earlier, I described how the raster scan technique works. Although you can't see it explicitly in the pour you just made because of the way Allegro graphically creates it on the computer screen, the effect is readily observable by changing the uniform trace width. Access this parameter by choosing **Shape→Global→dynamic params…** This will bring up the *Global Dynamic Shape Parameters* dialog having four tabs as shown. Select the *Void controls* tab now.

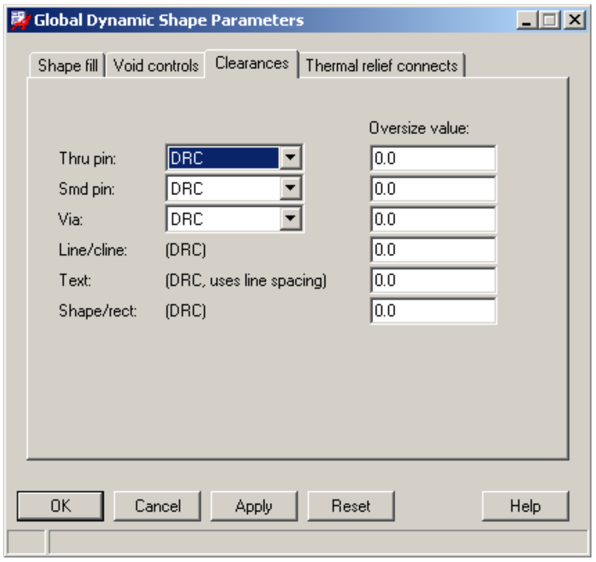
Note the *Minimum aperture for gapwidth* parameter is set to 4 mils. This means Allegro will choose apertures for draws necessary to realize 4 mils of detail resolution around poured objects. Note this detail pertains to voids. 4 mils happens to be Allegro's default, but it can be changed if we want. The following sequence of screen-shots clearly show how the voids are affected by this parameter. As the gap width resolution gets bigger, resolvable detail decreases:


10mils 25 mils 50 mils 75 mils 100 mils

Notice especially the third case, where the gap width is now 50 mils and the resulting draw aperture is too wide to fit between the padstacks. When the gap width is 75 mils observe that that areas of isolated copper appear to the right of the padstacks. These electrically disconnected regions are called *islands*, and typically serve no design purpose whatsoever. Always try to minimize their formation by appropriately moving things around when possible; usually this means traces rather than footprints. This example, of course, is artificial, since I'm using the extraordinary case of an absurdly large gap width to obviate their appearance, and merely decreasing the width will eliminate them. Generally, though, if you intend to use copper pours as an electrical net, unintended islands can break the pour's wanted connectivity, thereby defeating the purpose of connecting nodes. After increasing the gap width to 100 mils not even the islands fit! We conclude from this series of examples that there isn't much difference between 4 and 10 mils, and that large gap widths result in very poor spacial resolution. This reinforces my earlier comment about not making this parameter less than 4 mils unless you have very good reasons to the contrary.

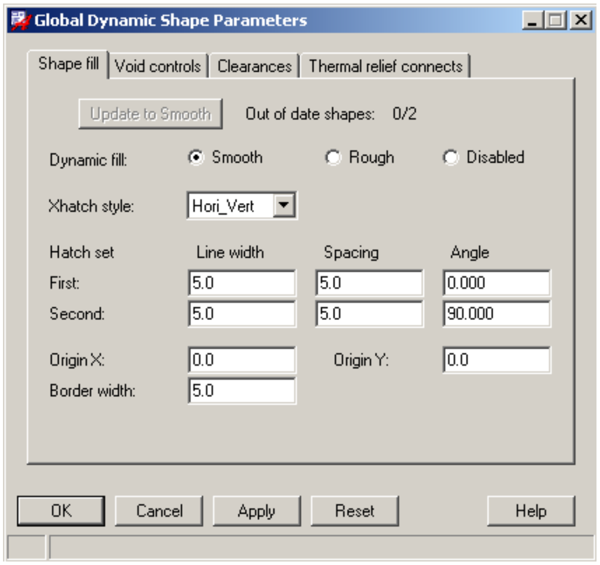
Notice that *Suppress shapes less than* is set for 25 squared mils. Try setting this to 100 squared mils with the gap width at 75 mils, and notice the islands disappear. This setting is typically used to automatically void small islands having areas smaller than this setting area setting. The remaining parameters you can experimentally investigate yourself. Note the *Artwork format* should be set for **Gerber RS274X**. I will will delay discussing this particular parameter until we get to the end of the lab.

The **Clearances Tab**, shown below, provides ways to specifically increase voiding for the six relational cases shown. The pull-down lists has two choices: *DRC* or *Thermal/anti*. I've chosen the default to be DRC and specified that *Oversize value* to be 0 mils. This means whatever is currently specified in the Constraints' settings, 8 mils in our case, will be used for autovoiding (and for the DRC as well). We can increase this by entering non-zero values in the Oversize value column for each of the six clearance types. The screen-shots above all show 8 mils of autovoiding for Cline's and Thru pins. Try increasing the Thru pin oversize by 5 mils and watch what happens. Since we haven't yet poured around any SMD pins, or text or adjacent shapes, you can't see the effect of oversizing these yet – but you will be able to soon. Be sure to reset any of these to 0 before going on.



		Oversize value:
Thru pin:	DRC	0.0
Smd pin:	DRC	0.0
Via:	DRC	0.0
Line/cline:	(DRC)	0.0
Text:	(DRC, uses line spacing)	0.0
Shape/rect:	(DRC)	0.0

The last tab, **Shape fill** controls two things: properties associated with dynamic filling, and how the shape is actually created. Everything below and including *Xhatch style* are settings to control only how cross-hatched [8] areas are created. I found the fastest way to explore these cross-hatching parameters is to create one and experiment with the settings on this tab.



Out of date shapes: 0/2

Dynamic fill: [✓ Smooth] Rough Disabled

Xhatch style: Hori_Vert

Hatch set	Line width	Spacing	Angle
First:	5.0	5.0	0.000
Second:	5.0	5.0	90.000

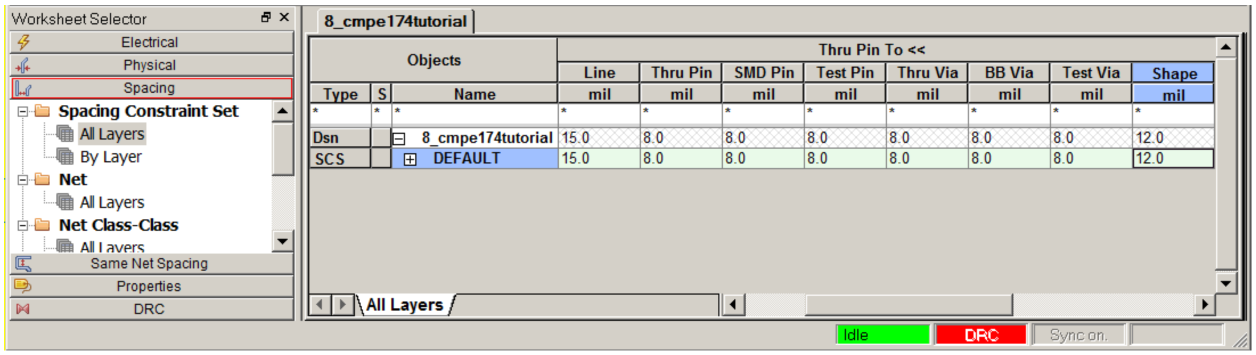
Origin X: 0.0 Origin Y: 0.0

Border width: 5.0

The button at the top, *Update to Smooth* and the three *Dynamic fill* radio buttons control how dynamic copper shapes are updated. Smooth gives the most accurate resolution, Rough takes far less time to update and Disabled suspends all dynamic updating until the button is pressed. These options mean very little in our project, but for really large layouts, the refresh time can be significant. Setting Rough will give a quick reasonable approximation of things, whereas Smooth though taking longer, is obviously preferred. Note that if you make any changes to the Constraints' table that affects any already poured autovoiding dimensions, Allegro will suspend dynamic autovoiding until you press the *Update to Smooth* button. Right now the button is inactivated, but when one or more shapes are "out of date", it won't be. Also, during that time when shapes are out of date, many normally autovoided regions will appear to be filled.

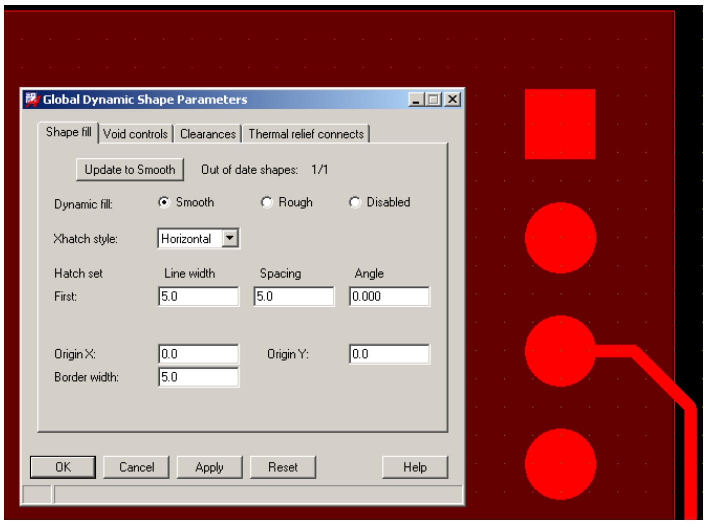
[8] These choices are only available when a shape is created, and are found on the Options' panel; see earlier discussion on pg. 40.

I'll do one example to show you how this works. Access the Constraints' manager, change the *Pin to Shapes* constraint from 8 mils to 12 mils, and watch what happens to the existing dynamic shapes' previously autovoided regions around affected pins. Expand the relevant column fields by first double-clicking the "Thru Pin To <<" field; (I've already done this). After changing the field, the DRC annunciator will turn red meaning it's now out of date. A screen-shot of which field you need to change is shown following:

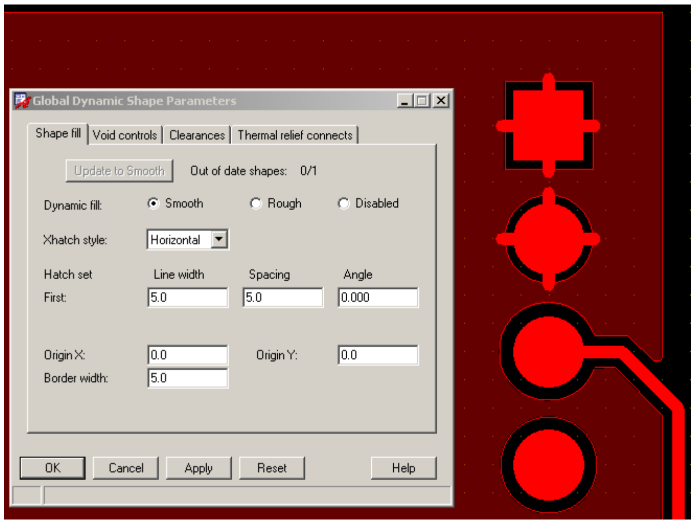


Objects			Thru Pin To <<							
			Line	Thru Pin	SMD Pin	Test Pin	Thru Via	BB Via	Test Via	Shape
Type	S	Name	mil	mil	mil	mil	mil	mil	mil	mil
*	*	*	*	*	*	*	*	*	*	*
Dsn		8_cmpe174tutorial	15.0	8.0	8.0	8.0	8.0	8.0	8.0	12.0
SCS		DEFAULT	15.0	8.0	8.0	8.0	8.0	8.0	8.0	12.0

Notice that as soon as you do this, all the autovoids now appear to be flooded. Actually, they're just out-of-date. Bringing up the *Shape fill* dialog confirms this. I've positioned the dialog next to the now filled padstack to clearly show you this relationship. The left image shows 1 out of 1 (we only have one shape currently on the design) shapes are out of date. Pressing the *Update to Smooth* will immediately re-establish dynamic re-pouring, as shown on the rightmost image (try it):



Out of date shapes: 1/1
Dynamic fill: [✓ Smooth] Rough Disabled
Xhatch style: Horizontal
Hatch set: Line width / Spacing / Angle
First: 5.0 / 5.0 / 0.000
Origin X: 0.0 Origin Y: 0.0
Border width: 5.0



Out of date shapes: 0/1
Dynamic fill: [✓ Smooth] Rough Disabled
Xhatch style: Horizontal
Hatch set: Line width / Spacing / Angle
First: 5.0 / 5.0 / 0.000
Origin X: 0.0 Origin Y: 0.0
Border width: 5.0

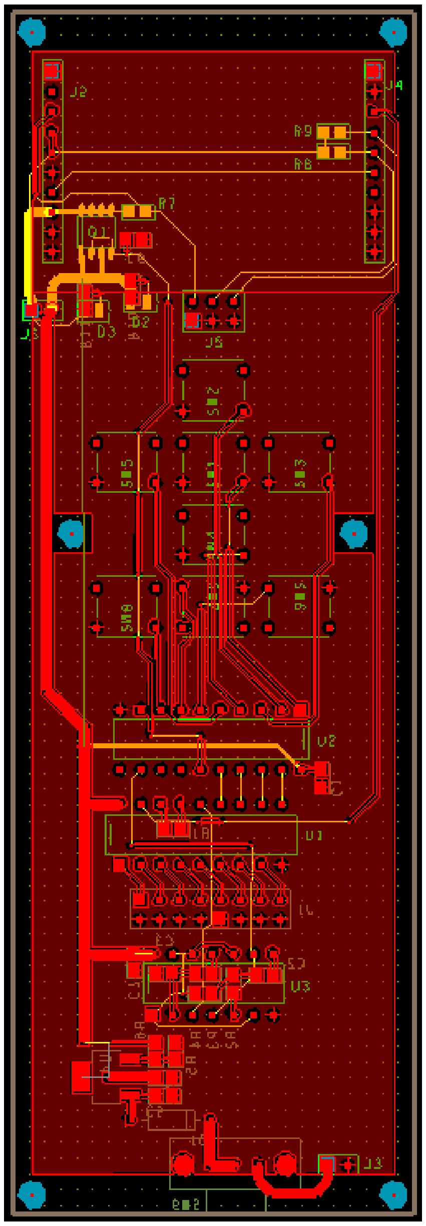
Notice, only those pins that aren't tied to the GND net show the new 12 mil voids; the others are simply re-poured as before. Do you remember why? Reset the constraint parameter back to 8 mils before proceeding.

What about creating these dyamic shapes piecemeal? That is, arranging them side-by-side with tangent edges so as to realize a larger equivalent area? So long as adjacent dynamic copper shapes have the the same net name, Allegro will seamlessly join them without autovoids; otherwise, isolation voids will necessarily appear based on current spacing constraints' settings (in this case 8 mils for *Shapes* to *Shapes*).

Editing Copper Pours

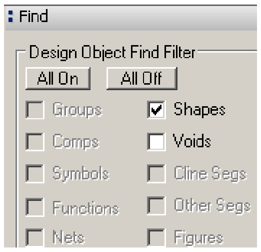
Refer once again to the color handout showing the red copper pour as a polygon on the bottom layer. Continue by creating the remaining irregular area below the top region you just poured beginning with a single bounding rectangle, and then edit that rectangle to adjust its' perimeter to look like the handout. Even though the following discussion uses a named dynamic shape, the editing techniques I'll show you apply to any shape.

First, why the irregular shape? The simplest answer is: *copper balance*. Board houses typically give us a bad time if the copper areas on both sides of a PCB aren't reasonably the same. The reason is that large boards with copper planes on only one side, and relatively little on the other, strongly tend to curl or warp. On small boards, like ours, this really isn't a problem. When I first ran this board, I placed a single large rectangular copper pour on the bottom layer and specifically instructed the board house to "run the board regardless of copper balance." This saved me getting an email or a phone call about the matter. On the board you're laying out, knowing how to edit shapes is a necessary skill, so I've made it part of the layout by having you minimize the copper area to improve the copper balance. On some boards, we resort to placing copper pours on the scant side (top in our case), tied to no net (the "dummy net" in the net choices) that effectively deposit large islands to improve the copper balance. Another way is to use *cross-hatching*, since this takes less copper than solid pours.

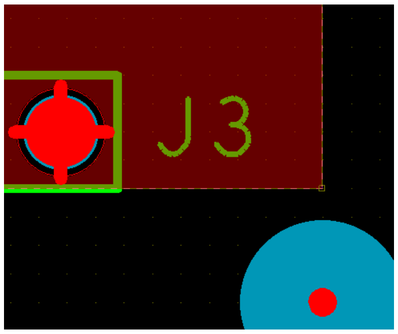


The left image shows the second dynamic copper shape, also tied to the GND net, just below the one made earlier. This was added using a temporary shape grid of 50 mils., making it easy to define the four corners of the whole rectangle. I also turned on the ASB view, since it had the board outline visible, but turned off all the top etch features in the Visibility panel. After creating the shape, I turned on the AST_ASB view specifically for the screen-shot shown.

Editing this basic rectangle is an easy skill to learn, and one well-worth knowing. Shapes can be edited in either General or Etch Edit mode. I like to use Etch Edit. But simply setting the filter for Shapes and then selecting the shape of interest won't work; rather, you must always begin by first selecting it from menu sequence **Shape → Select shape or void**, or use the faster icon found on the Shape Toolbar. The context will change and the Find panel will now restrict the selection set to only shapes and voids as shown. For this example, I've selected only for shapes. Once you've done this, then select the particular shape you want to edit, in this case the lower one just added.


Find
Design Object Find Filter
All On | All Off
[ ] Groups [✓ Shapes]
[ ] Comps [ ] Voids
[ ] Symbols [ ] Cline Segs
[ ] Functions [ ] Other Segs
[ ] Nets [ ] Figures

Once the shape you want to edit has been selected, its highlight will change but, most importantly, *small squares will appear around every vertext* point. In our case, the rectangle will have only four such vertexes, one on each corner. Note that vertexes do not appear around any poured areas. Positioning the mouse anywhere along the shape's edge causes the mouse icon to change to a double-arrow; clicking will create a vertex that can then be dragged to a new position. Or, you can click and drag any existing vertex. The example left shows what the small square vertex point looks like. Vertexes can also be deleted by first selecting them and choosing *Delete* from the right-click context sensitive menu. Remember too the Options panel will automatically change to the appropriate sub-class (in this case bottom) where you can then reassign nets and set the temporary *Shape grid* resolution.

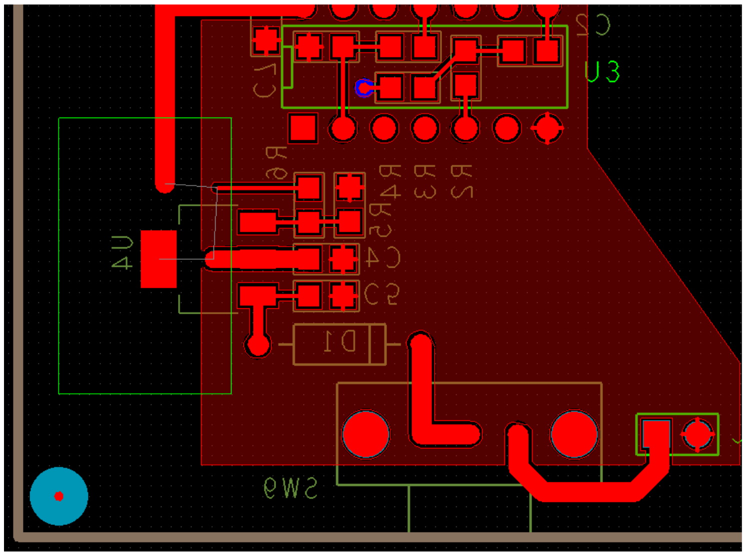


Each dynamic shape is independent of any other. Whenever you select a shape to edit, as you've just done, any voids and clearance changes made to it using the Global Dynamic Parameters dialog will *only apply to that particular shape*. This feature adds great versatility.

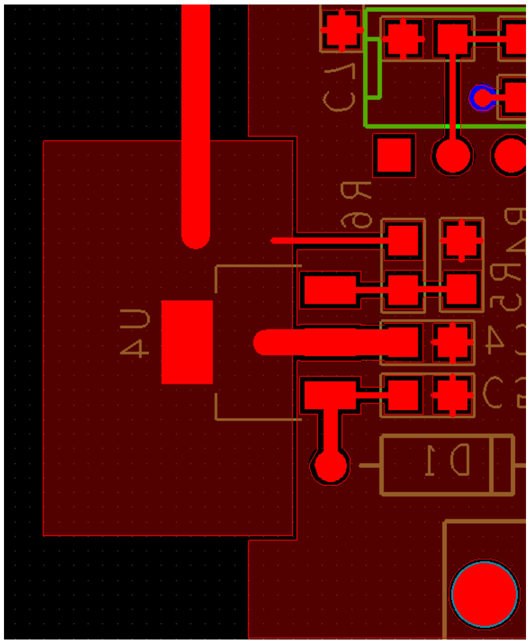
Proceed now to go ahead and edit the lower copper pour to look like the handout. You might have noticed there is another isolated copper area beneath the voltage regulater, U4. At this point, ignore this shape since we haven't yet placed it. We'll do that next.

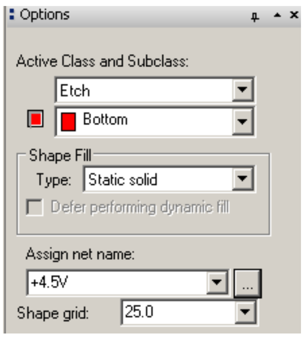
Copper Areas

The handout shows a rectangle of isolated copper poured beneath the LM317 linear regulator, U4. The SOT-223 package requires this part to be soldered to a thermal area that will lower its thermal resistance and hence its operating temperature. Generally, the thermal resistance varies linearly with area and should be computed before creating the dissipation area (this information is typically found as part of a device's datasheet)[9]. The detail below



shows the pad provided by the footprint for pin-4 is too small to be very effective, so we'll lay a larger copper area directly over it, effectively increasing the pad size. This implements a classic *heat-sink*. These kinds of pours are generically called *copper areas*, and Allegro refers to them as *static solid shapes*. Here, I've shown the heat-sink's perimeter with a green rectangle I temporarily created on the dimension layer to clearly show you what the board looks like before the static solid shape is added. Once the new copper area is created and tied to the +4.5V net, it will automatically connect to the two dangling traces, causing the associated ratsnest to disappear. We'll do that now.





Options

Active Class and Subclass:

Etch

Bottom

Shape Fill

Type: Static solid

Defer performing dynamic fill

Assign net name:

+4.5V

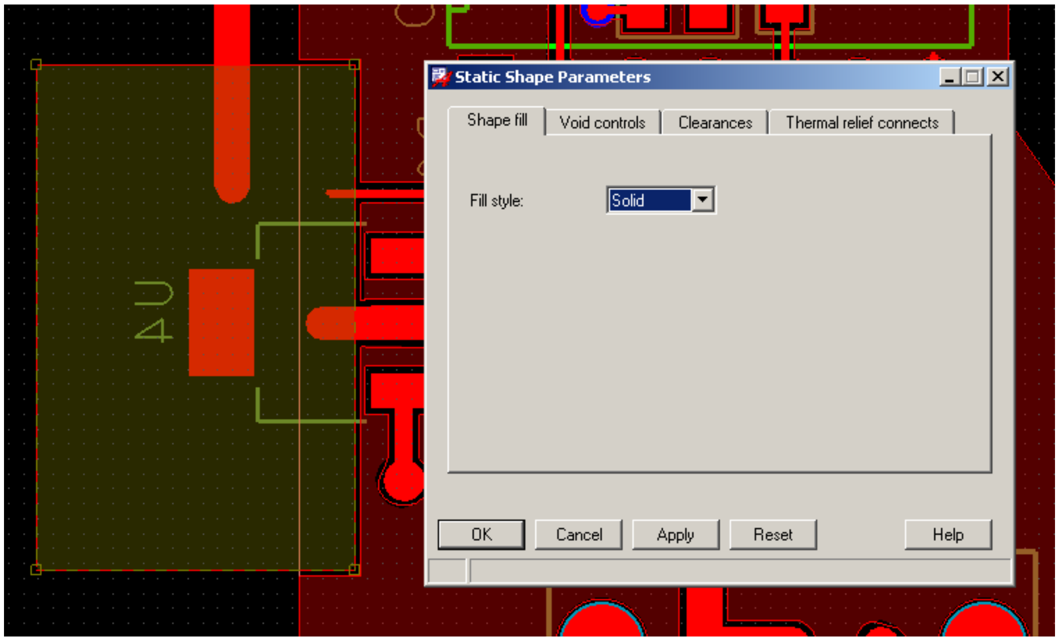
Shape grid: 25.0

Do this exactly the way you did earlier, but instead of choosing Dynamic copper, choose the Shape Fill to be *Static solid* instead. Don't forget to tie this to the "+4.5V" net.

Copper areas, unlike copper pours don't automatically create voids around poured objects, but like copper pours, they are also electrical objects and should be tied to a particular net for proper use. Also, notice there is no voiding inside this area, which we expect, but there is an 8 mil void around the perimeter to electrically insulate the regulator from the GND net. This void is a consequence of the Dynamic copper shape, not the new static shape. You might confirm from the schematic and datasheet that pins 2 and 4 are tied together as outputs.

[9] Refer to Audio/Radio Handbook, sec 4.14.1, NSC 1980; *Heat_Flow_reference.pdf* in your Appnotes folder.

A set of dialogs specifically for selected dynamic and static shapes can also be accessed. For example, first select the current static shape (using the menu sequence or icon), then from the context-sensitive menu choose *parameters*. This will bring up the *Static Shape Parameters* dialog shown below. You should have enough background at this point to explore these parameters yourself. Notice the vertex editing points on the copper area. These are used the same way as for the dynamic shape you edited earlier. Had you selected a dynamic shape instead, you would have accessed the *Dynamic Shape Instance Parameters* instead. Although these two dialogs appear to be identical, they aren't. The *Fill style* has only two options for copper, while copper areas have many more.

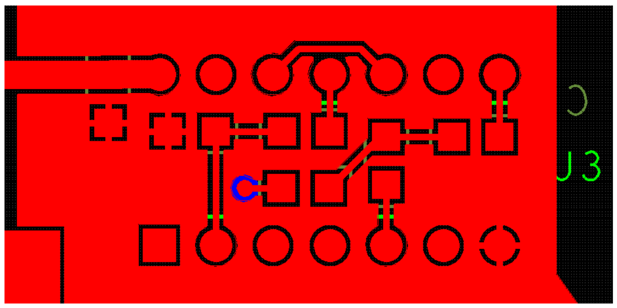


Fill style: Solid

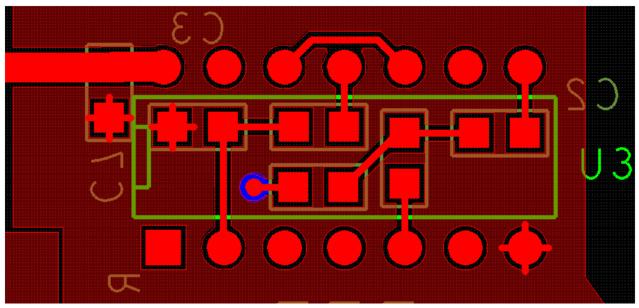
Be very careful using copper areas. Unlike copper pours, they won't automatically add voided safety regions between nets different that their own (in this case +4.5V). This will, however, cause a DRC flag to appear alerting that two nets are physically shorted, and you would find it if you're diligent about tracking down DRC errors (and you should; more about this later).

Anti-Copper Areas

Concentrate for a moment on the several 0805 SMD parts placed beneath U3. Observe the areas between the 0805 footprints have been filled and poured around with 8 mil automatic voids. This can make placement and soldering of associated parts physically difficult, making these connections prone to unwanted solder bridges. This isn't typically a problem when an enamel soldermask layer is present, but it is for engineering prototypes that have deliberately omitted this layer. Why would we want to do this? The answer is simple. During the design and prototyping phase, omitting the soldermask on early board runs often makes the board easier to modify, since we almost always end up doing just that. Final board revisions meant for production runs, however, will always be full-featured and include soldermasking. Since the PCB you're laying out is meant as an engineering prototype, we will mitigate the solder bridge problem by placing *anti-copper* areas around objects we don't want the pour to "pour" into. Incidently, this also indirectly improves the copper balance. This creates large mandatory voided areas, in this case for SMD parts, thereby making them easier to work with. The two screen-shots below should make this point vividly clear.



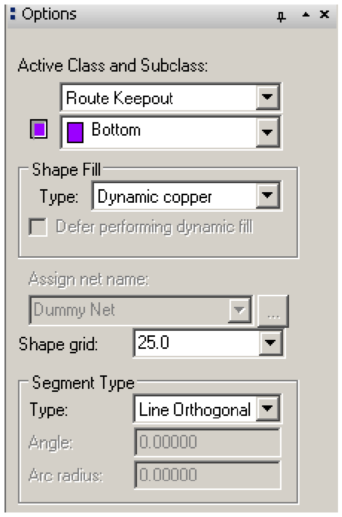
100% shape transparency.



39% shape transparency.

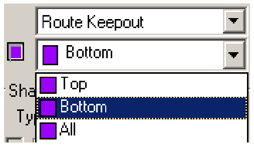
For the image on the left, I changed the shape transparency to "solid" (100%), while the right has the normal 39% transparency. Note these are both top views, the way the parts look while projecting down through the top layer. Obviously, when actually working with these parts, it's necessary to turn the board over causing everything to "mirror"; this is the reason the bottom-side lettering appears backwards (but you already knew this, right?).

Allegro calls anti-copper regions *keepouts*, and classifies them by what objects they "keep out"; in our case *shape keepouts*. We'll add one now beneath U3. Choose menu sequence **Setup** → **Areas** → **Shape Keepout** [10]. This changes only the Options' panel, as shown:



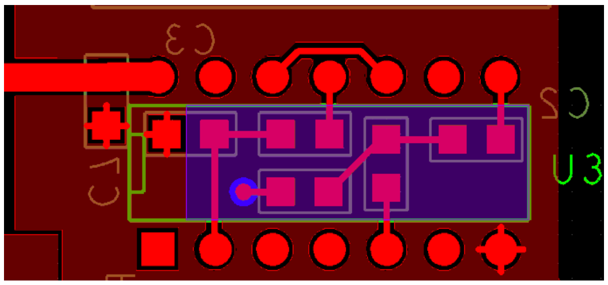
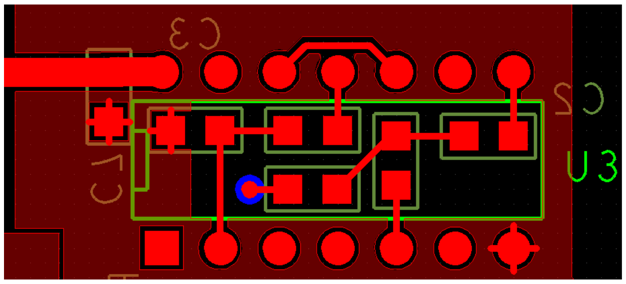
Active Class and Subclass:
Route Keepout
Bottom
Shape Fill
Type: Dynamic copper
Defer performing dynamic fill
Assign net name:
Dummy Net
Shape grid: 25.0
Segment Type
Type: Line Orthogonal
Angle: 0.00000
Arc radius: 0.00000

Allegro will choose the *Route Keepout* active class automatically, but you must specify everything else. Options in the pull-down list will show each of the design's etch layers (only top and bottom in our case) and *All*. These choices specify which layers the keepout will apply to. For example, in our case choosing *All* would cause any copper pours (or areas) on either the top or bottom sides to be voided. In this particular design, it wouldn't matter whether *Bottom* or *All* was selected, since no pour is used on the top layer. But for those cases where we might place a pour on the top (and they do exist for shielding and EMI purposes) we definitely would restrict the keepout only to the bottom layer. Setting the *Segment Type* to orthogonal makes it easier to draw, since all we want to create is a simple rectangle. The *ShapeFill* type doesn't seem to matter; it works equally well whatever choice is selected. This might be an architectural oversight due to a code re-use issue, or there might actually be some subtle differences – in any case, I haven't been able to find any.

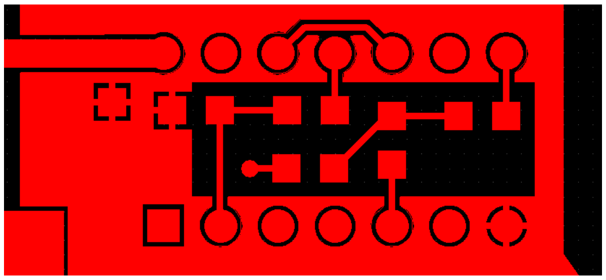


Be sure the sub-class is visible, then go ahead and add the keepout. As you draw the rectangle, be sure to visit each of the four corners sequentially, since Allegro merely appears to give you a finished rectangle after visiting two corners; think of these as "construction lines". When you reach the last closing point of the rectangle, the keepout will appear as a filled purple shape (with 39% transparency), as shown on the left-most image below.

[10] See pg. 39 for a screen-shot of the relevant menu list.

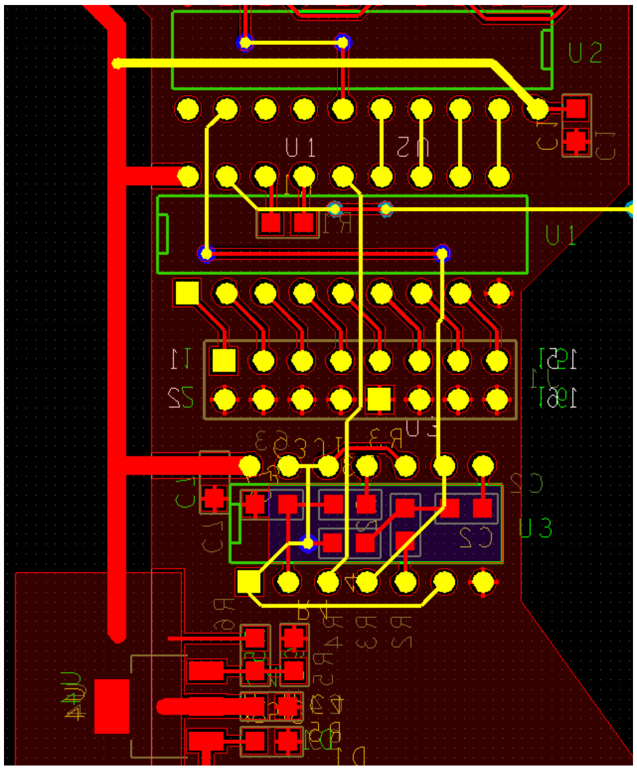



Turning off the new keepout (do this in the Options' panel), will result in the second image shown to the right. We can dramatically show only the solid copper to emphasize what the board will actually look like without documentation layers and 39% transparency.



The resulting keepout and surrounding copper pour is shown left. For this screen-shot, I set the shape transparency to 100%, selected the "Film: BOTTOM" view, and turned off everything on the top layer in the Visibility panel (try it).

4.8 Working With Text

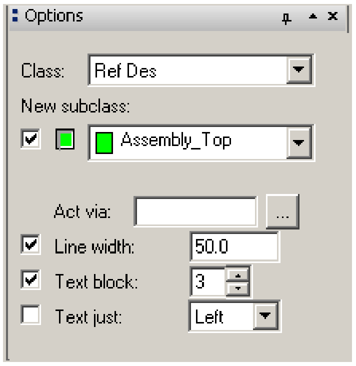


Text normally appears by default only on logical or documentation layers, like the silk-screen or assembly layers (others too), but it can also be deliberately placed by us on physical etch layers. Commonly this is done for PCB runs without solder-mask or silk-screen layers. Therefore, any wanted text can *only* appear on etch layers. Regardless of layer though, Allegro treats all text as non-electrical aperture draws having defined widths with specified heights. In this section, I'll show you how to both manipulate existing text and add new text. We'll begin by working with existing text.

You have most likely noticed, when all layers are visible (view set to *all_layers*), that many text strings appear duplicated on more than one sub-class. Indeed, the same thing is evident with certain line drawings, like footprint outlines; they also appear identically on assembly and silk-screen layers. However, our concern right now is with text. Shown left is a partial screen-shot with all layers having text visible (I turned off *Package Geometry / Place_Bound_Bottom* and *Place_Bound_Top* to de-clutter things a bit). Notice the mess of text that needs to be cleaned up! We'll do this by working with selectively visible layer combinations defined by how we *actually intend to use them*, since they really have different purposes; silk-screen goes to a board house and appears on the finished PCB, whereas assembly layers do not, meant as they are for self-documenting purposes. Obviously, this means they're not really all meant to be visible at the same time, making the *all_layers* view nonsensical – except to confirm the existence of particular objects and then work with them in their correct selectively visible context. We'll begin with the two assembly layers.

Cleaning Up the Assembly layers – The AST (assembly top) and ASB (assembly bottom) layers typically are not sent to the board house as part of the set of files needed to create a PCB. As I just noted above, these are largely for our use. The two working views, AST and ASB both have these assembly layers visible separately (the AST_ASB has both). Generally, I edit and move text around on these two layers as I go along. In this tutorial I waited until you first had some orienting experience before showing you how to do this. Begin by setting the *Views* in the Visibility panel for *File:AST*. When I created this colorview, I selected only one instance of the three occurances of reference designators for this set of layers, namely *Refdes / Assembly_Top* (find it and verify this).

Changing text size or layers. Text can be edited in any of the three master modes, but the best is *Placement Edit* [PLC]. Set this now, and filter only for text. Choose menu sequence **Edit → Change** (note there is no icon for this action). This will activate a set of change options in the Options' panel, as follows:


Class: Ref Des
New subclass: ✓ 🟩 Assembly_Top
Act via:
✓ Line width: 50.0
✓ Text block: 3
Text just: Left

These options cover more than just text, since the filter permits selecting and changing other objects. Relevant text settings are :

- **Class & New subclass** specifies where you want selected text moved and by checking the left-most checkbox. If the text is already on this layer, nothing happens.

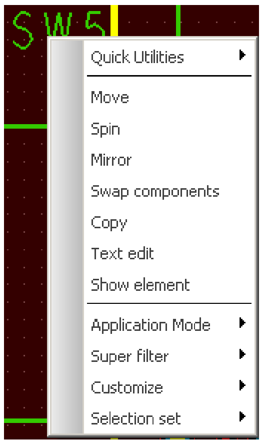
- **Text block** selects which line entry in the Text Table you want selected text to be sized to; in this example it's set for 3.

- **Text just** (justify) should be self-explanatory.

Whatever these three are set for will be simultaneously applied to any visible text you then select on the design. In this example, I only want to resize some text already on the AST layer that is perhaps too large (or small) and change it to whatever entry 3 is in the *Text Setup* table[11]. Try different sizes now and move text to the assembly_bottom (but move it back!).

Repositioning text is easy in *Placement edit mode*, since selecting symbols or text allows them to be moved about without explicitly having to set *Move* mode first.

> TIP: Since the Active class & sub-class doesn't have to always agree with where the text string is placed, provided the *New Subclass* box is left unchecked, an easy way to quickly get a finer grid to adjust text positioning is to select an etch class/sub-class.

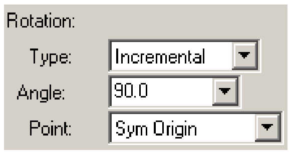


When we're are not in change mode (but still in PLC master mode with text filter set), hovering over any text string and pressing the right mouse button brings up a context sensitive menu with several options you will find very useful. The screen-shot left shows what happened when I did this for "SW5" on the assembly top layer. A summary of these command options follows below.

- **Move** is the same as actually selecting the string to begin with.

- **Spin** immediately allows you rotate the text object. Be aware of the three *Rotation* Options available in the Options' panel when spin is active:

[11] Refer to item 7. **Text Size** discussed earlier on page 15 with respect to parameter files.



Rotation:
Type: Incremental
Angle: 90.0
Point: Sym Origin

Experiment with these settings to see how they affect the spin operation. The first two are the most important. *Type* should be Incremental if you want the string to spin dynamically; otherwise, it only spins by whatever the angle is set for each time you select it.

- **Mirror** is used to *move* the text to another layer pre-programmed to be the “mirrored” layer; for AST this would be ASB. There is another closely related operation that allows us to mirror text *on the same* layer. This is most useful with the ASB layer where the text appears backwards because it’s really on the bottom-side of the PCB. See my comments below about *Mirror Geometry*.

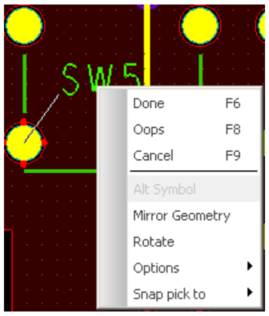
- **Copy** should be self-explanatory, but its use is subtle.

- **Text edit** accesses a dialog box allowing you to change the text.

- **Show element** brings up the usual prolix text box telling you everything possible about the string. This is most often used to identify what class & sub-class a particular string object is placed on, since Allegro’s pop-up text is next to useless.

Deleting Text requires that we explicitly first active *Delete* mode: choose **Edit → Delete** **,** then select the string with the mouse and click again to delete it. Notice that *both the Options’ panel and Find panel changes*. Often we have to tweak settings after entering delete mode. Remember, too, that once an object is deleted, Allgero remains in delete mode until told to explicitly exit (the usual right-click, Done… sequence).

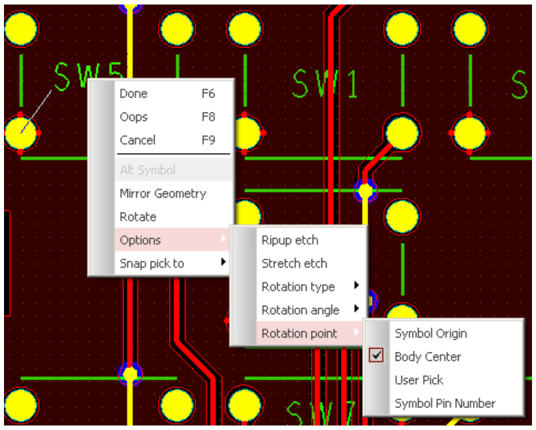
Selected Text Options. When text is actually selected, rather than hovering, and then bringing up the usual right-click context sensitive menu, a different set of options are available. The four relevant at the bottom are all quite useful. I'll comment on each in turn.



- **Mirror Geometry** does *almost* the same thing as *Mirror* does above. The difference is it allows you to make the text look like it would on the mirrored layer (top to bottom; or bottom to top) except it does it on the same layer. This is useful for showing reference designators correctly on the bottom-side without actually having to move from the ASB to AST layers to do it. Note the text merely *appears* mirrored, but is not actually moved to the mirror layer.

- **Rotate** essentially mimics the *Spin* selection above.

- **Options** essentially duplicates the same functionality as the Rotation dialog block on the Options' panel associated with the *Spin* selection above. The three parameters: Type, Angle and Point, now appear as menu selectable options. I've shown one of these menu sequences below:

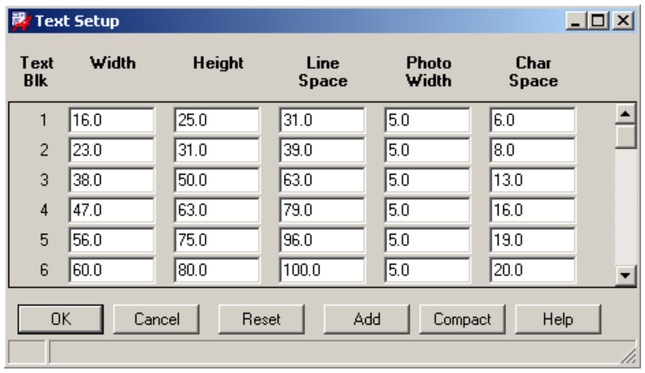


- **Snap pick to**. Experiment with these options to learn what they do. I have found that most of them ar not too useful.

Try playing with the Rotate, Mirror and Mirror Geometry commands to see their visual effects. Note that AST is green, ASB is brown, so it's easy to tell by inspection which respective layer text resides on. Go ahead and clean up both the AST and ASB layers now, sizing and arranging text to suit yourself. After gaining experience over several PCB designs, you will grasp more clearly how *you* want to use both the assembly top and bottom documentation layers.

Customizing Text

Before proceeding, you should learn how Allegro defines how all text appears. Earlier I noted that one of the tables imported from the parameter file at the very beginning of this lab[12], was *Text Setup*. Access this now to view the current text settings; it is found in the *Design Parameter Editor* : choose **Setup → Design → Parameters → Text tab → Setup Text Sizes.**. This will bringup the *Text Setup* dialog:



Text Blk	Width	Height	Line Space	Photo Width	Char Space
1	16.0	25.0	31.0	5.0	6.0
2	23.0	31.0	39.0	5.0	8.0
3	38.0	50.0	63.0	5.0	13.0
4	47.0	63.0	79.0	5.0	16.0
5	56.0	75.0	96.0	5.0	19.0
6	60.0	80.0	100.0	5.0	20.0

Shown is a subset of the 33 numbered entries in the table imported from the original parameter file. I recommend changing the *Photo Width* as follows:

#3	7
#4	8
#5	10
#6	10

This will thicken the text, making it more readable. Parameters on this dialog are discussed below:

Column headings:

- **Text Blk.** Identifies the text block by number. When adding text to a design, enter the number of the text block you want to use in the Options tab of the Control Panel.
- **Width** sets the width of each character.
- **Height** sets the height of each character.
- **Line Space** sets the distance between the bottom of the characters in one line to the top of the characters in the line below.
- **Photo Width** sets the width of the line used to photoplot and display text. This sets the actual aperture draw diameter used. If this is 0, the text *displays* at 1 pixel width in the workspace, but will *not* export to a Gerber "film" output file, since no draw can have a physical aperture of 0 (equiv. to no light!).
- **Char Space** Sets the amount of space between characters (kerning).

Buttons at the bottom of the dialog:

- **Reset** resets any parameter that may have been changed in the dialog box back to its previous setting. Note this is *only active in the current editing session.* After pressing OK, any new changes are accepted.
- **Add** adds a new text block to the dialog box below the default 16 blocks. Adding a text block disables the C Compact button.
- **Compact** removes *unused* text blocks and consecutively renumbers remaining blocks in both the dialog box and the current board design starting with 1. When Compact is first pressed, existing *Text Blk* numbers remain to reflect those text blocks beint retained, but the next time dialog box is opened, these same text blocks will then appear consecutively numbered, beginning with 1. If no unused text blocks exist, Compact is disabled. The Add and Compact buttons are mutually exclusive. Adding a text block disables the Compact button; compacting text disables the Add button.

[12] See the original discussion on pg. 15.

Cleaning Up the Silk-screen layers

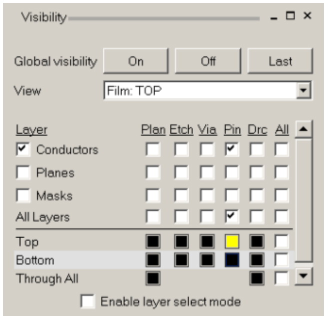
Two final silk-screen layers are intended to actually appear on finished PCB's: SST and SSB. Regardless of whether you intend to include them on a particular PCB board run, they should be cleaned up anyway, since they are often useful for hardcopy printed references. All layers eventually submitted to a board house are called by Cadence, *artwork*. Silk-screen artwork is applied as enamel paint to boards after everything else is done, and have traditionally been applied using silk-screen techniques. Consequently it's permissible to apply them over solder-mask (another enamel artwork layer) and copper etch objects alike. But any place solder-mask is voided from appearing, silk-screen ink will also be similarly prohibited. Some board houses allow us to specify solder-mask without silk-screening to save on cost, but never silk-screen without solder-mask. Therefore, you should always think of silk-screen artwork and the underlying solder-mask as being together. Since silk-screen text is going to appear on a real PCB, it must be sized so it's really printable; this means not making it too small and being sure the line width is at least 5 mils. Larger lettering is often easier to see by increasing the line width (aperture draw diameter), but this aspect is entirely up to us.

Before working with text on the silk-screen, we need to turn on which combination of the seven sub-classes we want to use. Below I've summarized relevant sub-classes noted earlier for the single classic top-side silk-screen[13].

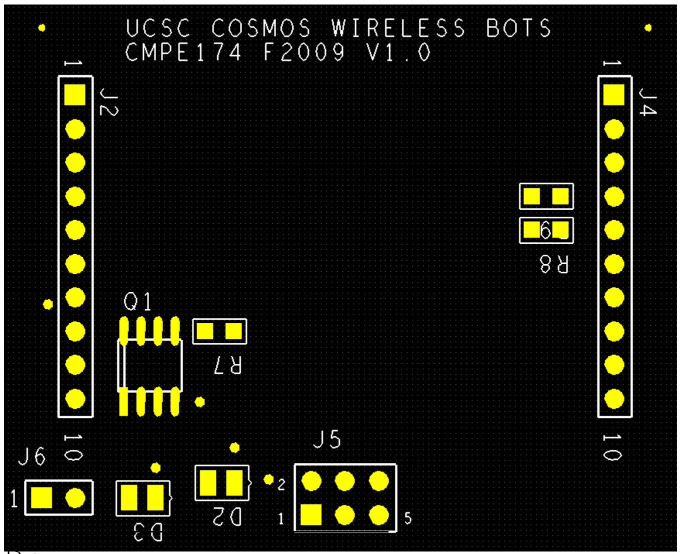
Top-side silk-screen:	Some combination, one or more of the following:
	Board Geometry / Silkscreen_Top
	Component Value / Silkscreen_Top
	Device Type / Silkscreen_Top
	Package Geometry / Silkscreen_Top
	Ref Des / Silkscreen_Top
	Tolerance /Silkscreen_Top
	User Part Number / Silkscreen_Top

These seven layers with seemingly identical information gives us great latitude in selecting the ones we want to include when finally exporting the finished design to a board house, or to use for particular purposes. Here, we'll combine the three logical layers indicated in bold and underlined. Using the Color Dialog, turn off everything; then, use the Options' panel to find the three sub-classes just noted and make them visible. As you do this, note what these sub-classes visually contain. After rendering these three visible, make the only the pins visible on the top layer.

Just in case you forgot how to do this, I've included a quick screen-shot to remind you. If you're still not comfortable using these checkboxes, now is a really good time to experiment with all of them, since selective visiblility really helps while editing silk-screen text. The reason we want pins and vias visible, is because placing text on padstacks isn't very useful. Allegro will merely blank the overlap anyway (won't print it directly on padstacks: pads, pins or vias).

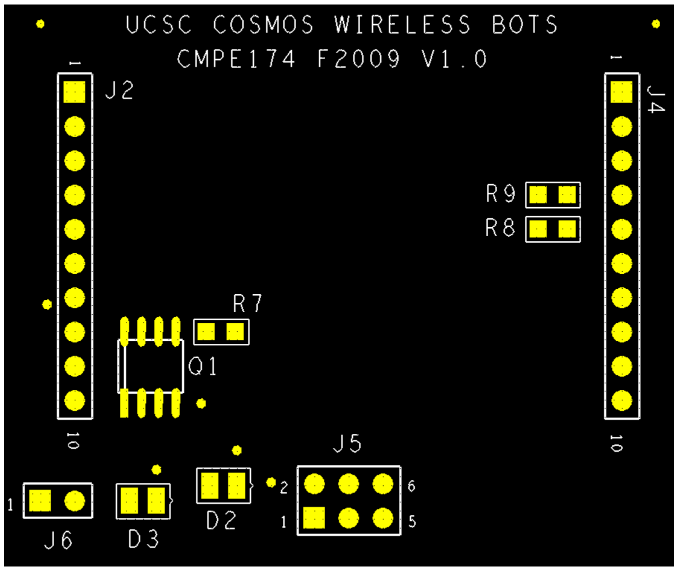


Layer	Plan	Etch	Via	Pin	Drc	All
✓ Conductors				✓		
Planes						
Masks						
All Layers				✓		
Top	■	■	■	🟨	■	
Bottom	■	■	■	■	■	
Through All	■				■	



A portion of what you should see in the upper-left region of your design is shown left. The text at the top we'll add in just a moment. Arrange the reference designators and make the pin numbers for J2, J4 and J5 smaller (use #2 from the text table). Also add the missing number "6" to J5; somehow this didn't get included when the footprint was created!

13 See pg. 11.



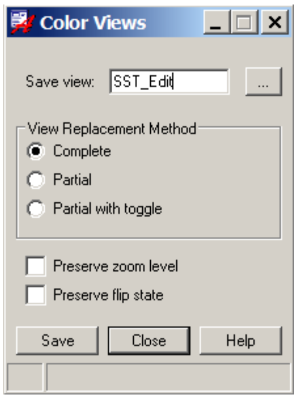
The result after making these changes is shown left. Note that if you now make the top etch visible, that some of the lettering will appear over traces. This makes no difference, since the silk-screen is applied over the *solder-mask*, which is usually dark green or blue, making the etched copper trace beneath difficult to see anyway. Also notice that I didn't place "Q1" inside the SOIC's rectangle, since after mounting the chip you would no longer be able to see it. Although silk-screen is certainly useful for assembly, it's also useful when finding parts and pins numbers on a fully assembled PCB for troubleshooting! By contrast, it's quite permissible on the AST view to place reference designators, like "Q1" in the middle of the part.

As you progress to clean up the text on these combined SST layers, you may notice errors, like J1 has its' pin numbers mirrored but the text is actually on Board Geometry / Silkscreen_top; be sure this is moved to the bottom layer where it belongs, and change their size to #2.

Adding Text

Add the board documentation text at the very top of the board. Adding text is easy, just follow menu sequence **Add** → **Text** or use the *add text* icon. After entering the context, set the *Text block* size to "3" in the Options' panel. Be sure the active class & sub-class are correct.

Since this set of visible layers is very useful, let's save it as a new entry in your view file set. Do this by entering the command: "colorview create" (or type "helpcmd" and select the command).

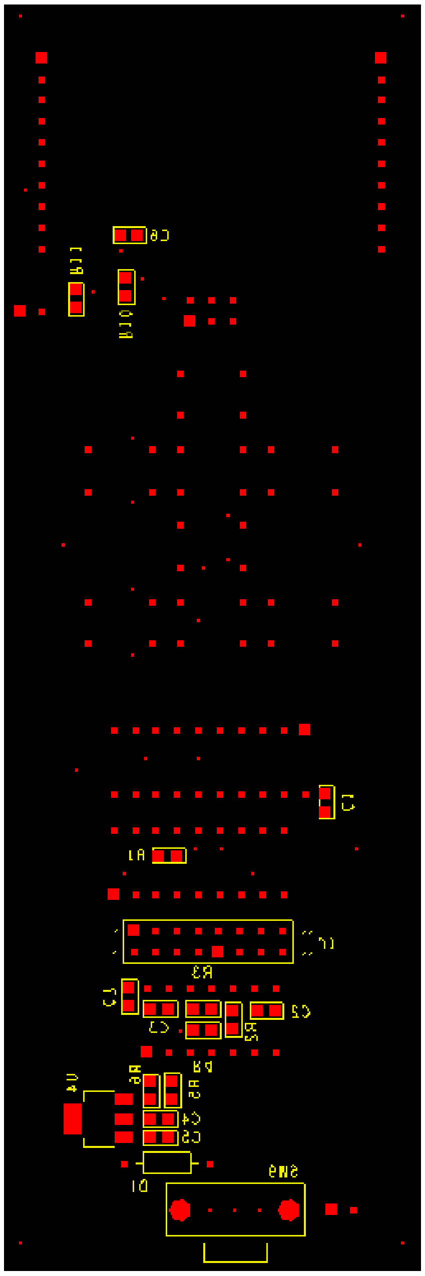

Color Views

Save view: SST_Edit

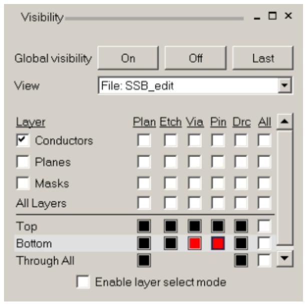
View Replacement Method
- (●) Complete
- ( ) Partial
- ( ) Partial with toggle

- [ ] Preserve zoom level
- [ ] Preserve flip state

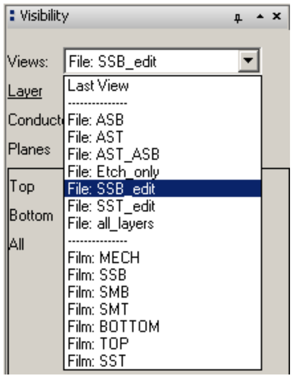
Once the *Color Views* dialog comes up, name the new file as shown. Pressing Save will write the file to your current working directory, namely where your current active *.brd file is located (this should be ..\Handset\pcb), and also immediately add it to your Views list. Just press *Save*.



After you've completed this for the top SST layers, repeat the whole process for the bottom-side of the board and similarly add a new colorview for the bottom SSB layers, named "SSB_edit". Your finished SSB_edit view should result in something similar to the screen-shot left with the setup in the Visibility Panel below.



This is similar in all respects to what you did on the top-side of the board.

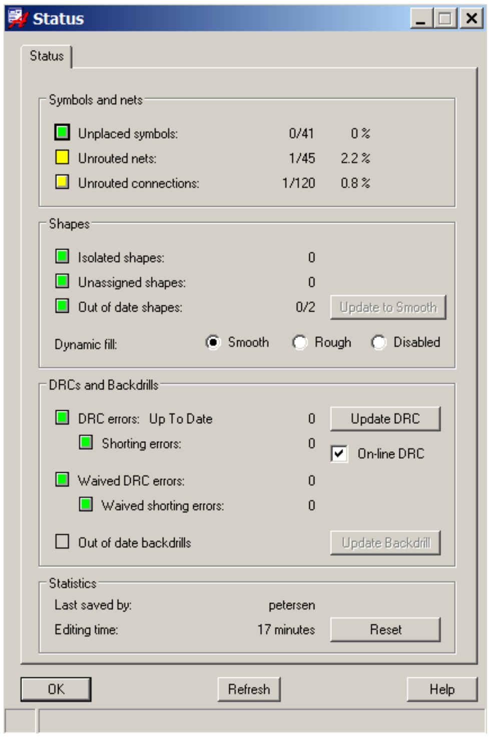


At this point, your views' list should now include the two new colorviews useful for viewing and editing the two composite silk-screen layers. For reference, I've included a screen-shot showing this.

4.9 Checking For Errors

Checking Statistics

Working with large designs having many nets and layers makes it easy to overlook some. Allegro provides a means to check the progress of things by displaying a set of useful statistics. Generally, this can be consulted anytime, but it's really most useful at the end of the layout process. Choose menu sequence: **Display → Status…**This will bring up the *Status* dialog shown.



Status

Symbols and nets

🟩	Unplaced symbols:	0/41	0 %
🟨	Unrouted nets:	1/45	2.2 %
🟨	Unrouted connections:	1/120	0.8 %

Shapes

🟩	Isolated shapes:	0	
🟩	Unassigned shapes:	0	
🟩	Out of date shapes:	0/2	Update to Smooth

Dynamic fill: [● Smooth] ○ Rough ○ Disabled

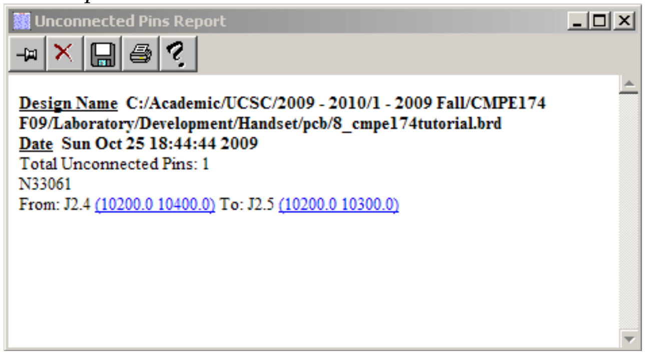
DRCs and Backdrills

🟩	DRC errors: Up To Date	0	Update DRC
🟩	Shorting errors:	0	[✓ On-line DRC]
🟩	Waived DRC errors:	0	
🟩	Waived shorting errors:	0	
☐	Out of date backdrills		Update Backdrill

Statistics

Last saved by:	petersen	
Editing time:	17 minutes	Reset

This single-tabbed dialog reports on the status of three major areas shown. Before proceeding "all boxes must be green", but two are showing yellow – equivalent to a warning; we could proceed but at our own risk. The difficulty shown, a very common problem, is to believe we've routed everything, but in fact Allegro says we haven't. According to the report, there is one unrouted net and one unrouted connection. The best way to find it is to turn all sub-classes off, but make sure the ratsnest is visible. Then scan the board for any unrouted rat lines. It will definitely be there, but may sometimes be very hard to find. Allegro can also help you. Just press the yellow annunciator *Unrouted connections*. This will run the *Unconnected Pins Report* as shown:

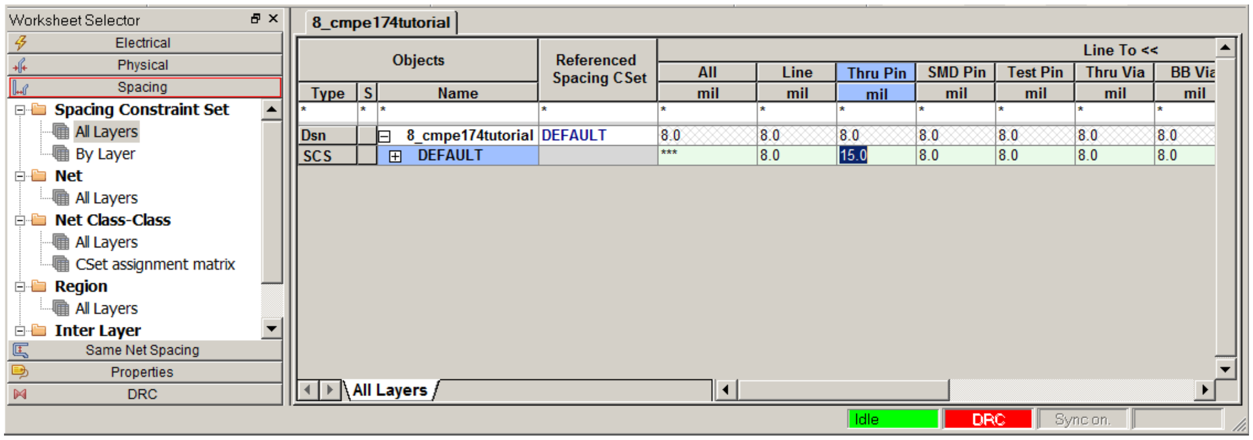


Unconnected Pins Report

Design Name C:/Academic/UCSC/2009 - 2010/1 - 2009 Fall/CMPE174 F09/Laboratory/Development/Handset/pcb/8_cmpe174tutorial.brd
Date Sun Oct 25 18:44:44 2009
Total Unconnected Pins: 1
N33061
From: J2.4 (10200.0 10400.0) To: J2.5 (10200.0 10300.0)

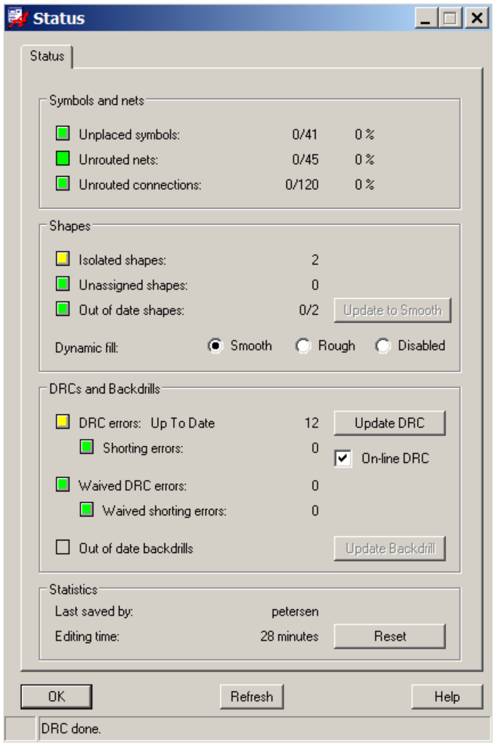
This tells us the net involved and which pins it connects. Pressing the blue highlighted coordinates will place the associated pin in the center of the screen. So zooming in very close will rapidly help track things down. Note that pressing any of the other colored buttons, except *Unrouted nets*, similarly generates a report.

Whether you have errors or not, lets make some to review how to troubleshoot and show how Allgro flags DRC them. Access the Constraint Manager as shown and first expand the "Line to <<" field resulting in all of the possible "Line To" spacings (I've already done this below). Then increase the *Line to Thru Pin* spacing from 8 mil to 15 mils on the Constraint's Manager spreadsheet, as shown; once changed notice the DRC turns red in the lower right corner to remind us to run it and bring it back up to date:



Objects			Referenced Spacing CSet	Line To <<						
				All	Line	Thru Pin	SMD Pin	Test Pin	Thru Via	BB Via
Type	S	Name		mil	mil	mil	mil	mil	mil	mil
*	*	*	*	*	*	*	*	*	*	*
Dsn		8_cmpe174tutorial	DEFAULT	8.0	8.0	8.0	8.0	8.0	8.0	8.0
SCS		DEFAULT		***	8.0	15.0	8.0	8.0	8.0	8.0

Close the Constraint’s Manager and access the *Status* dialog again. Notice the DRC errors now shows red. Press the *Update DRC* button to bring the status check back up-to-date. It should turn yellow flagging us that DRC errors still exist.



Status

Symbols and nets

🟩	Unplaced symbols:	0/41	0 %
🟩	Unrouted nets:	0/45	0 %
🟩	Unrouted connections:	0/120	0 %

Shapes

🟨	Isolated shapes:	2	
🟩	Unassigned shapes:	0	
🟩	Out of date shapes:	0/2	Update to Smooth

Dynamic fill: [● Smooth] ○ Rough ○ Disabled

DRCs and Backdrills

🟨	DRC errors: Up To Date	12	Update DRC
🟩	Shorting errors:	0	✓ On-line DRC
🟩	Waived DRC errors:	0	
🟩	Waived shorting errors:	0	
☐	Out of date backdrills		Update Backdrill

Statistics

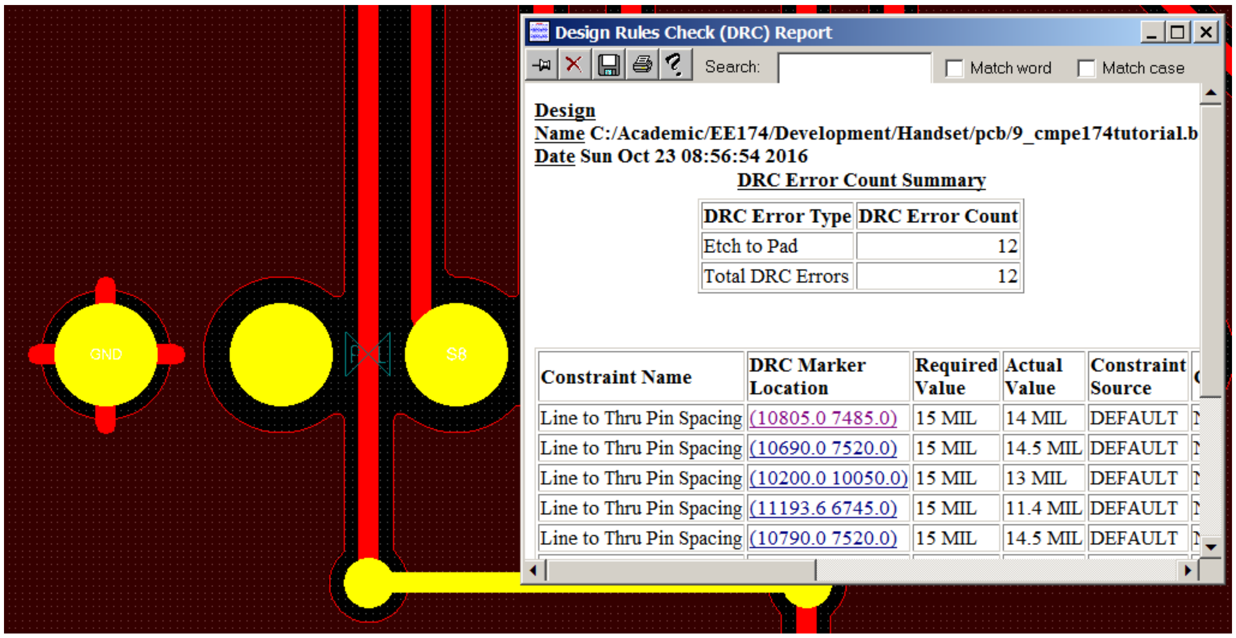
Last saved by:	petersen
Editing time:	28 minutes

DRC done.

Notice the DRC now shows 12 errors (yours may differ, since this is strongly layout dependent).

The fastest way to track these down and view where they are is to first zoom in very close (anywhere), then generate the *DRC errors* report by pressing the yellow button. The report should agree with the 12 reported errors and provide an exhaustive, detailed, listing about each one. Find the column titled *DRC Marker Location* and notice that each entry is highlighted in blue; these are selectable hot-links. Clicking any of them immediately places that particular DRC error in the center of the screen. An example of this is shown below.

Allgro flags DRC errors with “bowties” having two single-lettered flags inside each “bow” to indicate the type of error. In this example I clicked the second lined error. Allegro tells us this ia a *Line to Thru Pin Spacing* error with a required value of 15 mils, but having an actual value of only 13 mils. Notice the bowtie shows this as a “PL” error, shorthand for pin-to-line. Had this been a line-to-line error, the bowtie would read “LL” instead. Since we deliberately introduced two different types of errors, go down the list and find a *Line to Thru Pin Spacing* error and observe how Allegro flags it.



Design
Name C:/Academic/EE174/Development/Handset/pcb/9_cmpe174tutorial.b
Date Sun Oct 23 08:56:54 2016

DRC Error Count Summary

DRC Error Type	DRC Error Count
Etch to Pad	12
Total DRC Errors	12

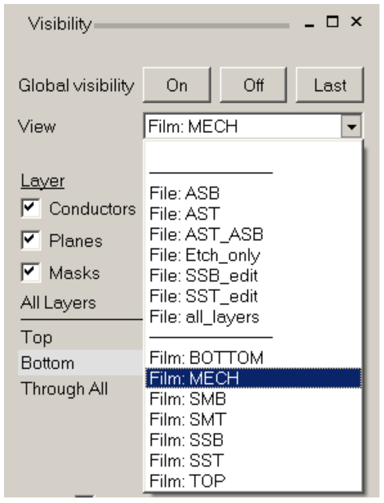
Constraint Name	DRC Marker Location	Required Value	Actual Value	Constraint Source
Line to Thru Pin Spacing	(10805.0 7485.0)	15 MIL	14 MIL	DEFAULT
Line to Thru Pin Spacing	(10690.0 7520.0)	15 MIL	14.5 MIL	DEFAULT
Line to Thru Pin Spacing	(10200.0 10050.0)	15 MIL	13 MIL	DEFAULT
Line to Thru Pin Spacing	(11193.6 6745.0)	15 MIL	11.4 MIL	DEFAULT
Line to Thru Pin Spacing	(10790.0 7520.0)	15 MIL	14.5 MIL	DEFAULT

This method of finding errors is very powerful. Since we had the online DRC activated earlier, had you violated any of these spacing constraints, Allegro would have generated bowties. This probably didn’t happen though, since Allegro’s autorouter enforced these rules dynamically as you worked at routing traces manually. Certain errors, however, could have occurred, such as footprints having overlapping package keepouts. Be sure you’ve set the two constraints back to 8 mils and proceed to finish checking your design for problems. All errors and warnings must be resolved before proceeding on to the next phase.

4.10 MANUFACTURING

This final "manufacturing" phase essentially reduces to creating a particular set of artwork and drill files to be used by a boardhouse to manufacture our PCB. Particular because boardhouses have specific requirements that must be rigorously met regarding artwork and drill file formats that must be fully understood before proceeding. Naturally, then, this requires a good working familiarity with the boardhouse you intend to submit these files to. For this laboratory, we'll be submitting to Alberta Printed Circuits (APC), a Canadian boardhouse very popular for short-run engineering prototypes. APC offers several 2-layer services: *Basic* and *Plus Prototypes*. Basic is the least expensive, features no solder-mask or silk-screen and turns in 1-day. Plus is more expensive, allows solder-mask, and selectively optional silk-screen on the top, both top and bottom, or neither; cost is adjusted accordingly. Plus also has pre-screening and layout integrity check that Basic does not. Originally, this project was submitted as a Basic board run with no solder-mask or silk-screen. I'll show you how to prepare files for either type.

Up to this point, we have used entries found in the *Views'* list without distinguishing between those grouped under *File* and those grouped under *Film*. Since we're about generate output files that will be sent on to a board house, we need to discuss what these groups mean. The screen-shot below shows all current selections, both File and Film views.



File views represent working visible layer combinations frequently used in the Design Workspace. These were first saved in *colorview* files (*.color) for later reuse. Recall we loaded these during one of the initialization steps at the beginning of the lab[14] and recently we added two more while editing text on the silk-screen layers.

Film views represent visible layer combinations that will eventually be used to create files to be sent on to a boardhouse. Cadence calls this step the *artwork generation phase*, and calls the output *artwork files*. Both of these terms, *artwork* and *film* are firmly rooted in the history of PCB manufacture. The lithographic process employed to make PCB's originally involved the use of inter-negative photographic film, where the content of Gerber Photoplotter files contained the "artwork" instructions necessary to expose and produce the inter-negative "film" images. Even though boardhouses may no longer use photographic film, the traditional terminology still remains widely used today. Remember, these film views are saved and imported in parameter files.

PAY ATTENTION TO THE BOARD HOUSE - Obviously, we can't proceed until we know something about APC. Go to their website now at http://apcircuits.com/ and *thoroughly read* over their requirements for both their Basic and Plus Prototype services. This is always the first step: *learn what the boardhouse wants before proceeding*. You won't understand everything at this point, but it will give you the "big picture", and having done so will elucidate references I make to particular points found on their site.

APC Basic Prototype – The following points summarize what APC minimally wants us to know. When reading this, note that APC calls Cadence *artwork* files *Gerber* files, where the latter is more commonly encountered in industry, but you will often see them equivalently used. In fact, among these three equivalent terms, artwork, film and Gerber, "Gerber" is the most common and "film" the least.

1. They specify 7 mil design complexity, so our 8 mil Level B choice at the beginning is just fine. This used to be 8 mils, but they reduced it to make their service more competitive with other board houses.

2. They will only use FR-4 62.5 mil glass-epoxy PCB base material with 2 oz. copper plating. This is the classic, generic, 2-layer PCB material used everywhere; so we're happy.

[14] See earlier page 18.

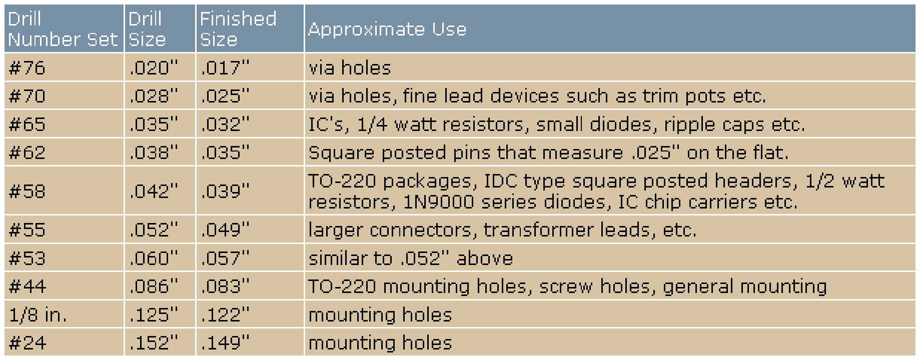
3. Five required files:
   - "**Board top side**": this Gerber file defines the Top etch or copper layer that should also have identifying text "anywhere outside the board outline"; we will include this.

   - "**Board bottom side**": this Gerber file defines the Bottom etch or layer; identifying text as for the top side.
   - "**NC drill file**": this ASCII file contains NC drill coordinates for all drilled holes on our design and is intended to be input to automatic *numerically controlled* industrial XY drill machinery.

   - "**NC Tool list**": also known as the *drill rack*, this ASCII file contains a simple table associating tool numbers, used in the NC drill file, with actual drill diameters.

   - "**Board Mechanical layer**"[15]: this is a special artwork file we will need to create. APC further tells us that it must have the board outline and may show drill locations as a visual aid. Note the NC drill file actually contains the XY locations of various drills, so if we include them here it truly is just that – for visually noting drill target locations. We'll include the visual drill information, since it will also be useful to us as well.

2. APC gives us 10 preselected drill sizes for free. To avoid extra charges, we need to restrict our use of drills to these:



Drill Number Set	Drill Size	Finished Size	Approximate Use
#76	.020"	.017"	via holes
#70	.028"	.025"	via holes, fine lead devices such as trim pots etc.
#65	.035"	.032"	IC's, 1/4 watt resistors, small diodes, ripple caps etc.
#62	.038"	.035"	Square posted pins that measure .025" on the flat.
#58	.042"	.039"	TO-220 packages, IDC type square posted headers, 1/2 watt resistors, 1N9000 series diodes, IC chip carriers etc.
#55	.052"	.049"	larger connectors, transformer leads, etc.
#53	.060"	.057"	similar to .052" above
#44	.086"	.083"	TO-220 mounting holes, screw holes, general mounting
1/8 in.	.125"	.122"	mounting holes
#24	.152"	.149"	mounting holes

   Carefully understand the distinction between *Drill Size* and *Finished Size*. The former is the actual drill diameter, the latter accounts for the plating allowance[16], typically 3 mils. Note the "Approximate Use" column is just that: approximate; don't pay too much attention to it! Your design will have a few drills not found on this list, so they will need to be changed to avoid incurring extra cost. Prudent considerations must, of course, be weighed when deciding whether or not to change a drill. I have found from experience that these sizes are usually sufficient, and they certainly are in this design.

3. Board Routing is used around the perimeter of your board's outline to "cut out" the final PCB. The router bit size used is normally 94.5 mils in diameter; other smaller sizes or non-rectangular or square board geometries will incur extra charges. To save cost, APC tiles your particular PCB onto a much larger 2-layer board along with other customer's board submissions. At the end, they need to slice up the panel, breaking out each individual order, and they use routing to do this. Understanding this process will help us to properly place *Cutting marks* on the four corners of your board on the Board Mechanical Layer.

APC Plus Prototype – The summary points following add to the list above for the Basic Prototype service:

1. Four additional required Gerber files:
   - Artwork files for both top and bottom silk-screen layers.
   - Artwork files for both top and bottom solder-mask layers.

[15] APC no longer requires this, but it's good practice to include a mechanical layer to clarify questions they may have anyway – whether they use it or not; some board houses refer to this as the "Fab" layer and appreciate its' presence.
[16] See page 36, where this term was defined and discussed earlier.

2. Silk-screen is white, and any features (lines or text) must be at least 6 mils. Note that silk-screen layers are commonly referred to as *legend* layers in industry. Remember those 5 mil photo widths in the Text Table? If any of these appear on the silk-screen layers, they will need to be increased.

3. Green 3 mil dry film solder-masking is used on both sides of the board. This is sometimes referred to as *solder-mask over bare copper* (SMOBC).

4. FR-4 is still used, but we have a greater choice of thicknesses.

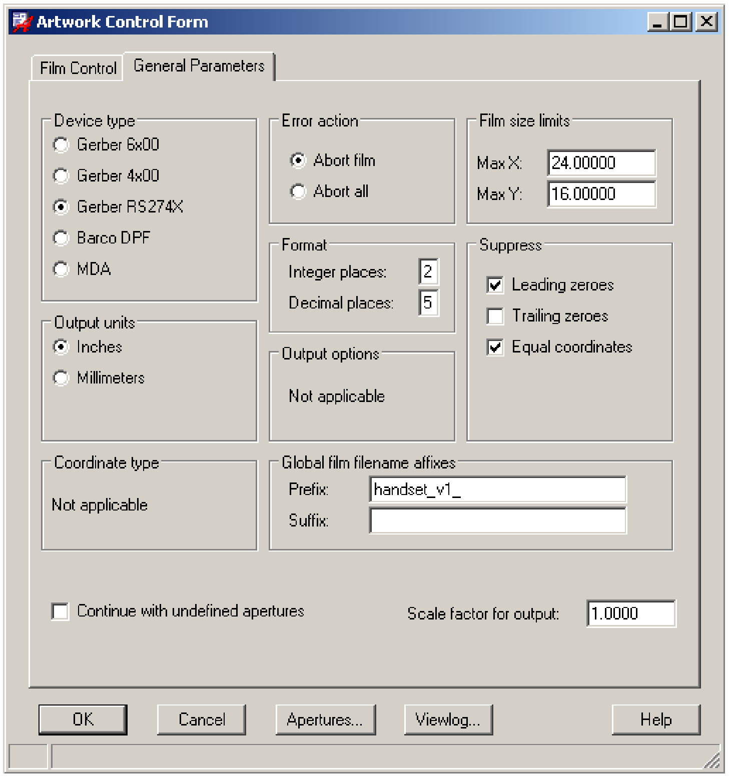
Additional APC Notes – The following summarizes additional things that apply to both service types:

1. Use the same measurement system with the same resolution throughout – both for Gerber files and drill files. We are requested to use an NC drill resolution of "2.4".

2. RS-274D or RS-274X formats[17] are acceptable. We will use the embedded aperture format, RS-274X.

3. We get a discount by using their online submission engine.

CREATING ARTWORK FILES - We'll begin by generating the necessary Gerber or film files. Choose **Manufacture ⟶ Artwork…** or use the faster icon. This will bring up the *Artwork Control Form*. Further select the *General Parameters* tab; we'll set these parameters first then go back to the other tab.


Artwork Control Form

Film Control | General Parameters

Device type: ○ Gerber 6x00 ○ Gerber 4x00 [● Gerber RS274X] ○ Barco DPF ○ MDA

Error action: [● Abort film] ○ Abort all

Film size limits: Max X: 24.00000 Max Y: 16.00000

Format: Integer places: 2 Decimal places: 5

Suppress: [✓ Leading zeroes] ☐ Trailing zeroes [✓ Equal coordinates]

Output units: [● Inches] ○ Millimeters

Output options: Not applicable

Coordinate type: Not applicable

Global film filename affixes: Prefix: handset_v1_ Suffix:

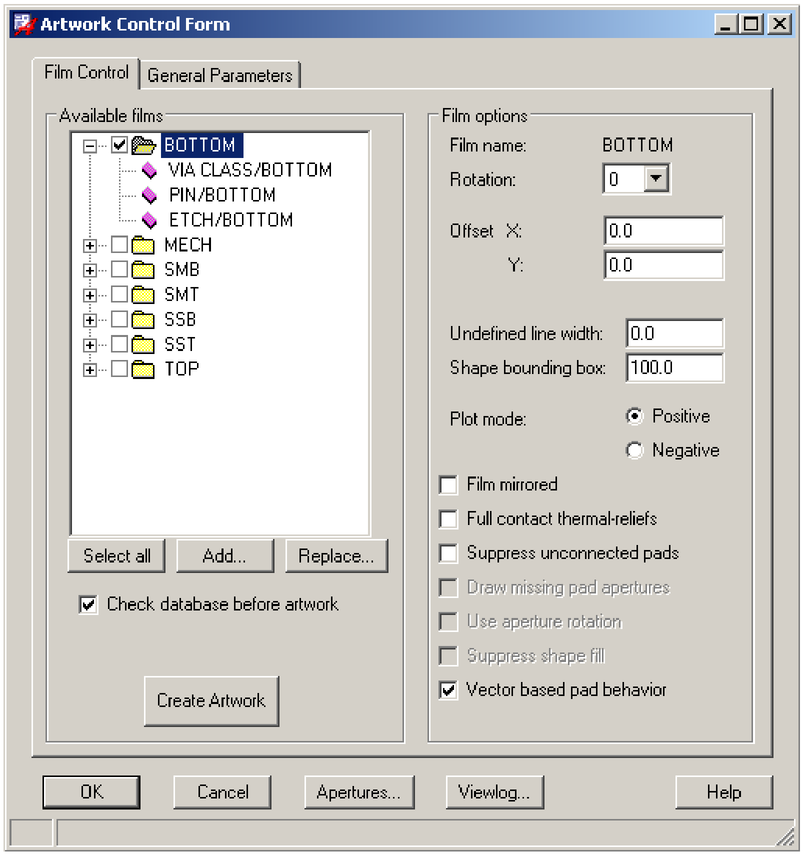
☐ Continue with undefined apertures

Scale factor for output: 1.0000

These settings apply only to generated artwork files. Set everything as shown. The *Global film filename affixes* allows us to define prefixes and suffixes to generated filenames. It's wise to always generate some identifying string in filenames. This is especially useful when you have a problem with the boardhouse and want to send them new files; bump the revision number and you'll avoid costly mixups. Pressing the *Help* button accesses a set of quick delineation of these various parameters.

[17] Ibid. footnote 3 on pg 13.

Select the *Film Control* tab. This is where each wanted film file is create, first as a stored record, denoted by the virtual folder icon, and second as a created artwork file. The screen-shot below shows that seven films are currently defined. These all came in with the technology template file (*.tcf) imported at the beginning of this laboratory. These artwork files constitute the minimum needed for most boardhouses.



Artwork Control Form

Film Control | General Parameters

Available films:
- BOTTOM (checked)
  - VIA CLASS/BOTTOM
  - PIN/BOTTOM
  - ETCH/BOTTOM
- MECH
- SMB
- SMT
- SSB
- SST
- TOP

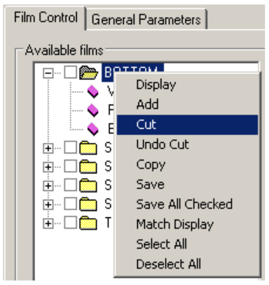
Check database before artwork: ✓

Film options:
- Film name: BOTTOM
- Rotation: 0
- Offset X: 0.0
- Offset Y: 0.0
- Undefined line width: 0.0
- Shape bounding box: 100.0
- Plot mode: [✓ Positive] Negative
- Film mirrored: ☐
- Full contact thermal-reliefs: ☐
- Suppress unconnected pads: ☐
- Draw missing pad apertures: ☐
- Use aperture rotation: ☐
- Suppress shape fill: ☐
- Vector based pad behavior: ✓

I have selected one of them, BOTTOM, to specifically show what sub-classes are to be included in the resulting artwork file. Check it now as shown and press the *Create Artwork* button. This will immediately create a file named *handset_v1_BOTTOM.art*, and write it to your *artwork* folder whose location was defined earlier in the *User Preferences Editor* [18]. Look now and be sure this file is actually present in your **…Handset/pcb/artwork folder,** then delete it. Remember, this folder is where all files relevant to the manufacturing phase are placed. Having them in a sub-folder off your working directory is a nice way to keep them segregated from the master board files.

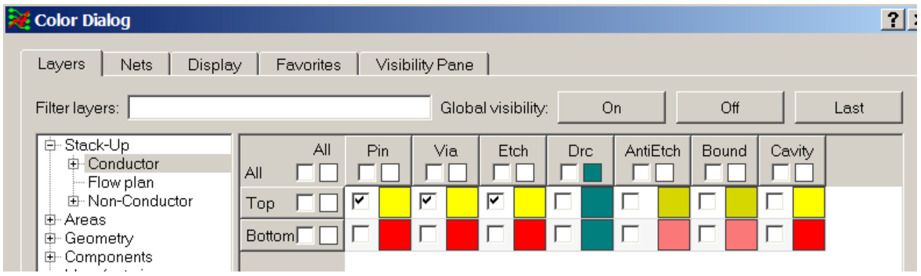
Since I want you to know how to create artwork files, we'll delete and re-create them now. We will also want to add one additional film file that isn't here already.

Delete them by highlighting each filmname (it doesn't matter whether its checked), right-click and select *Cut* from the context sensitive menu; this is shown left for the BOTTOM film record. Proceed to remove all the remaining films as well (one will always remain, so after later creating the first one it too can then be removed). Note that *unless you press OK*, the deletions won't stick.



[18] See pg. 8.

Creating New Film Views - Each new film record is created directly *from the sub-classes visible* at the time of its creation. Therefore, each new artwork film begins by first making all sub-classes invisible, then turning on only those we want in the new film. As an example, we'll do this now for the Top copper layer, which is defined by *only* the three usual etch classes & sub-classes checked as shown in the screen-shot following. The example uses the Color Dialog, but you could, of course, make the three visible by selecting them individually on the Options' panel, but it's much faster using the Color Dialog. A good way to see the effect of these three is to begin with everything off (Global Visiblity off), then additively select them one at a time.



	All	Pin	Via	Etch	Drc	AntiEtch	Bound	Cavity
All	☐	☐	☐	☐	☐ 🟩	☐	☐	☐
Top	☐	✓ 🟨	✓ 🟨	✓ 🟨	☐ 🟩	☐ 🟨	☐ 🟨	☐ 🟨
Bottom	☐	☐ 🟥	☐ 🟥	☐ 🟥	☐ 🟩	☐ 🟥	☐ 🟥	☐ 🟥

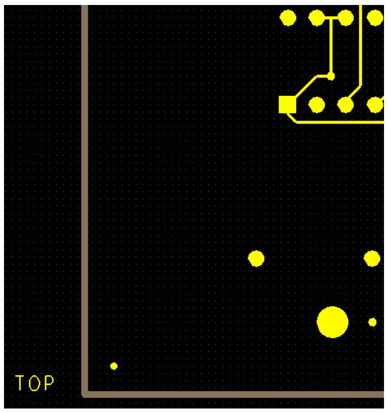
After you have set only these three sub-classes visible, open the *Film Control* panel and add a new film titled: **TOP**. If you just happened to have left this as your last film and thus weren't able to delete it, click the film's name and rename it. Then right-click anywhere in the *Available films* window and choose *Add*. If you have problems getting the menu to appear, select the single entry and right-click / Add. After successfully adding this new film it will immediately appear in the Views' list as *Film:TOP*, and anytime it's selected only the three sub-classes it defines will show in the design window. After you've learned how to do this, the list below shows all the films that need to be created (including the one you just did); add all of them now.

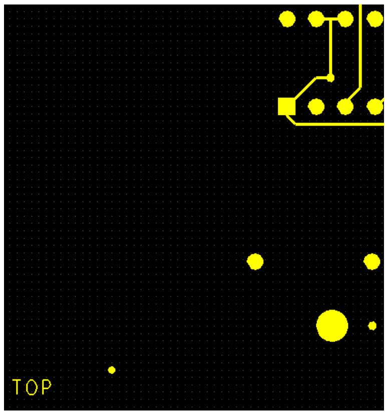
TOP (Top-side copper)	
ETCH / TOP (for traces)	Add APC text to this layer
PIN / TOP (for padstacks)	
VIA CLASS / TOP (for vias).	
BOTTOM (Bottom-side copper)	
ETCH / BOTTOM (for traces)	Add APC text to this layer
PIN / BOTTOM (for padstacks)	
VIA CLASS / BOTTOM (for vias).	
SMT (Top-side solder-mask)	
PIN / SOLDERMASK_TOP	
VIA CLASS / SOLDERMASK_TOP	
BOARD GEOMETRY / SOLDERMASK_TOP	Add APC text to this layer
SMB (Bottom-side solder-mask)	
PIN / SOLDERMASK_BOTTOM	
VIA CLASS / SOLDERMASK_BOTTOM	
BOARD GEOMETRY / SOLDERMASK_BOTTOM	Add APC text to this layer
SST (Top-side silk-screen)	
REF DES / SILKSCREEN_TOP	
PACKAGE / GEOMETRY / SILKSCREEN_TOP	
BOARD GEOMETRY / SILKSCREEN_TOP	Add APC text to this layer
SSB (Bottom-side silk-screen)	
REF DES / SILKSCREEN_BOTTOM	
PACKAGE / GEOMETRY / SILKSCREEN_BOTTOM	
BOARD GEOMETRY / SILKSCREEN_BOTTOM	Add APC text to this layer

Remember that APC wants us to include some layer text with these various Gerber files to clearly identify them. I've indicated which sub-class should be used in each film for this purpose. We'll do that next.

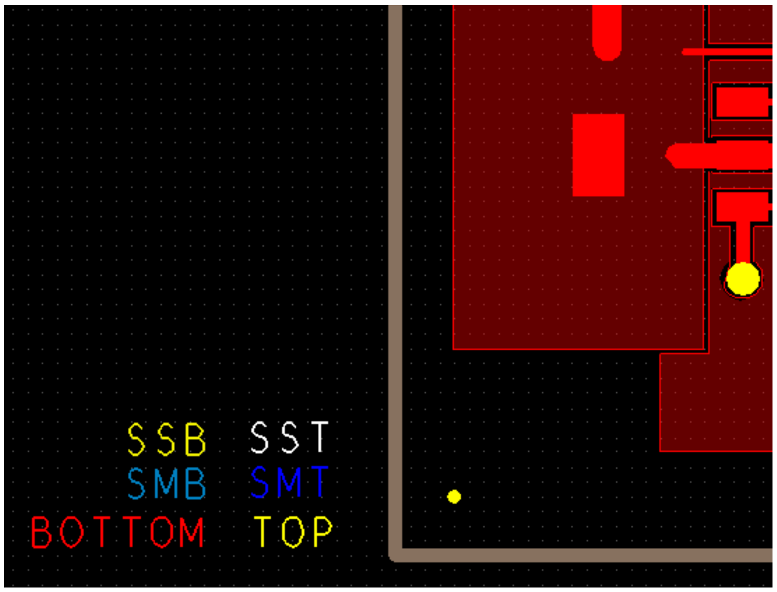
When you're finished with these new film records, do a sanity check by scrolling through them in the Visibility Panel to *make sure they look right*. Now, how do we add APC's requested text on these films? As an exampley, I'll show you how to do this with the TOP film view.

Begin by selecting the Film:TOP view and then turn on the board's outline (*Board Geometry / Outline*). Note that having the outline sub-class visible does *not* make it part of the Top film view, but it does help immensely in gauging where to place the new text. Down near the bottom left add new text "TOP" on the top etch layer. After doing this just reselect the Film:TOP view; the outline will disappear and the new text will be evident as shown by following two side-by-side screen-shots (I used #3 text size, but it doesn't matter; don't make it too small or large though).





Repeat this procedure for each of the remaining layers. If you're careful, text can be nicely arranged. Place text on the layers indicated in the list above. Notice I'm making use of the Board Geometry class to do this. After finishing, the new text identifiers should look something like the following:

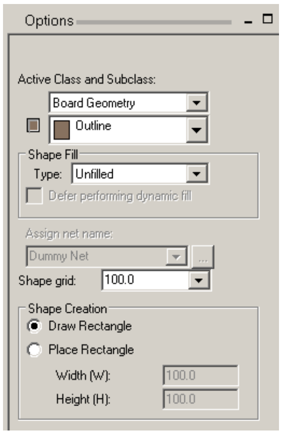


Remember, the film views *are for the boardhouse*, not us. Often they can be used while we're working on a layout, but they shouldn't otherwise be modified; if you need that, create a suitable file view and add it to the list. We'll create the mechanical layer next.

Adding a Mechanical Layer & Cut Marks

By now you know this reduces to blending particular sub-classes. We need to include the board's outline with cutting marks at the four corners and visual drill information.

We know which layer the board outline is on, so begin by making only the *Board Geometry / Outline* visible. Allegro has a very nice feature for adding cutting marks automatically, but it *only works with shapes*. Since we created your board outline using lines, the first thing to do is add an unfilled rectangular shape precisely overlaying the existing board outline. Unfilled shapes have only one very skinny line thickness (that's why we used the lines instead so we could specify their thickness). Do this now using the shape rectangle tool. Specify a *Shape grid* of at least 100 mils to avoid misaligning the new rectangle over the existing board outline, and be sure to specify the *Shape Fill* type as *Unfilled*. In the future, you can initially draw the board outline using lines to give it substantial visual thickness and also add an appropriate unfilled shape (a rectangle in this case) so cutting marks can be added later. After the rectangle has been added, proceed to add the cut marks.


Options

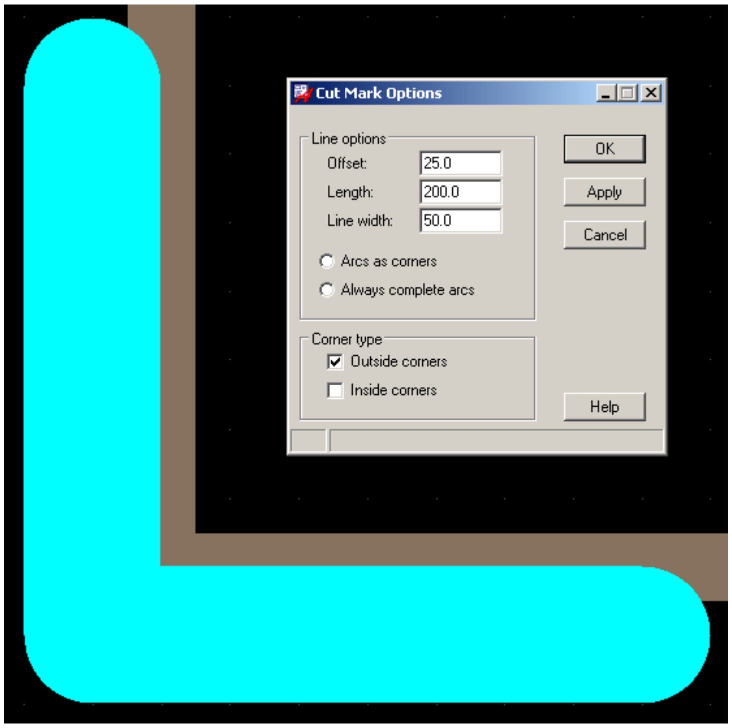
Active Class and Subclass:
Board Geometry
Outline

Shape Fill
Type: Unfilled
Defer performing dynamic fill

Assign net name:
Dummy Net
Shape grid: 100.0

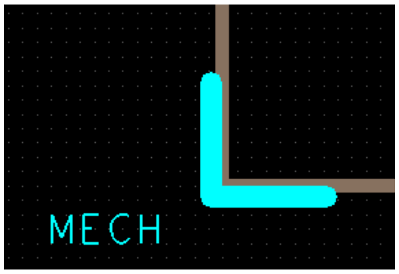
Shape Creation
(●) Draw Rectangle
( ) Place Rectangle
Width (W): 100.0
Height (H): 100.0

We *could* manually add these, but it's much easier to let Allegro do it for us. Choose menu sequence **Manufacture** ⟶ **Cut Marks** to access the *Cut Mark Options* dialog, and enter the *Line options* as shown below. Remember APC uses a 94.5 mil router, so specifiying a *Line width* narrower than this guarantees it will be completely ground away; therefore, I chose 50 mils. The length is not critical. I usually make at least four times the width, so the *Length* is 200 mils. The *Offset* is actually the most critical, since APC will align their router bit agains the *inside corner* edge, so the the cut marks must be increase by exactly half the line width to accomplish this. Hence the offset is 25 mils. After pressing the *Apply button*, Allegro places four outside corners at each vertex corner of the board outline. The screen-shot following summarizes what happens with a big blow-up of the lower-left vertex. Notice especially the cutting mark's inner edge precisely bisects the original thick line, whose center lies on the actual board dimension. Allegro places these cut marks on the *Board Geometry / Cut_Marks* sub-class; be sure this is visible! When you're done, add identifying text "MECH" to this sub-class in the lower left region, as shown in the second image to the right. This clearly tells APC (and us) this is the new mechanical layer.


Cut Mark Options

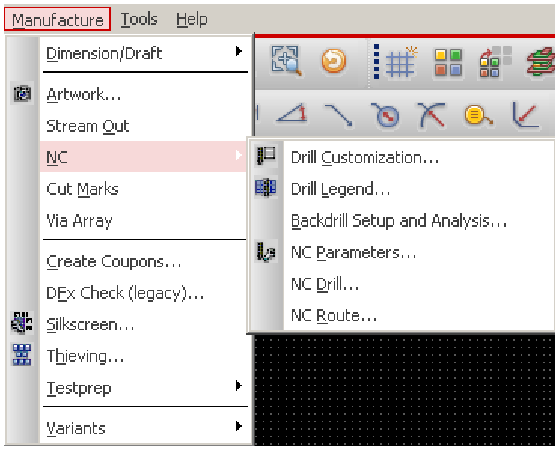
Line options
Offset: 25.0
Length: 200.0
Line width: 50.0
( ) Arcs as corners
( ) Always complete arcs

Corner type
[✓] Outside corners
[ ] Inside corners



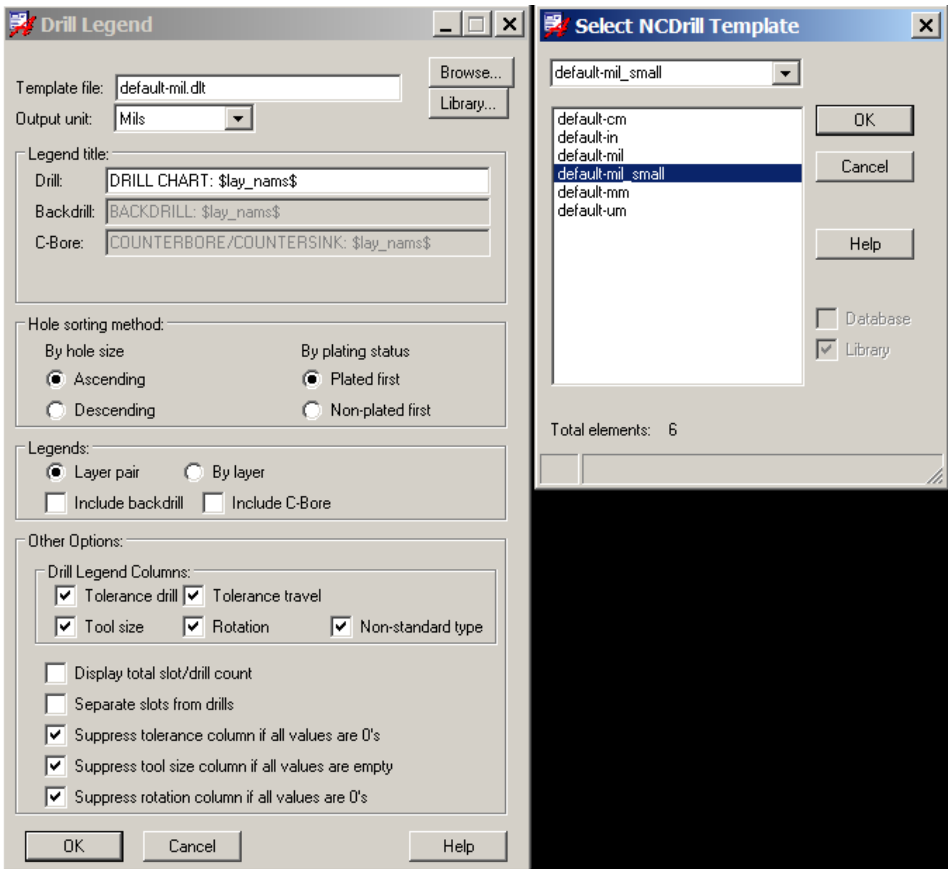
Adding a Drill Legend

The second feature we want to add to the new mechanical layer are visual drill targets and drill legend. This is easier to see than explain. Allegro creates this automatically and always places it on the *Manufacturing / Nclegend-1-2* layer; make sure this sub-class is visible before proceeding. Add the targets and legend by choosing menu sequence **Manufacture** ⟶ **NC** ⟶ **Drill legend…** I want to show you the menu options here, since we'll be using them next:



The six sub-menu options encompass all operations dealing with drill data. We'll make further use of several of these shortly to further generate the last two files containing drill information. Before doing that we need to inspect the state of our drills and possibly make some adjustments, and we'll do that by inspecting the drill legend.

When the *Drill Legend* dialog appears (below left), select the *Library* button to view available NCDrill templates. Select the one shown, **default-mil_small** and press *OK*. Allegro will specify it in the *Template file* field. This template is a variation of the Cadence *default-mil* script that otherwise results in a very BIG drill legend and should be in your …Handset/pcb/ folder. The file is actually named: default-mil_small.dlt.



Drill Legend

Template file: default-mil.dlt

Output unit: Mils

Legend title:

Drill: DRILL CHART: lay_nams

Backdrill: BACKDRILL: lay_nams

C-Bore: COUNTERBORE/COUNTERSINK: lay_nams

Hole sorting method:

By hole size: [✓ Ascending] Descending

By plating status: [✓ Plated first] Non-plated first

Legends: [✓ Layer pair] By layer

☐ Include backdrill ☐ Include C-Bore

Other Options:

Drill Legend Columns: ✓ Tolerance drill ✓ Tolerance travel ✓ Tool size ✓ Rotation ✓ Non-standard type

☐ Display total slot/drill count

☐ Separate slots from drills

✓ Suppress tolerance column if all values are 0's

✓ Suppress tool size column if all values are empty

✓ Suppress rotation column if all values are 0's

Select NCDrill Template

default-mil_small

default-cm
default-in
default-mil
default-mil_small
default-mm
default-um

☐ Database ✓ Library

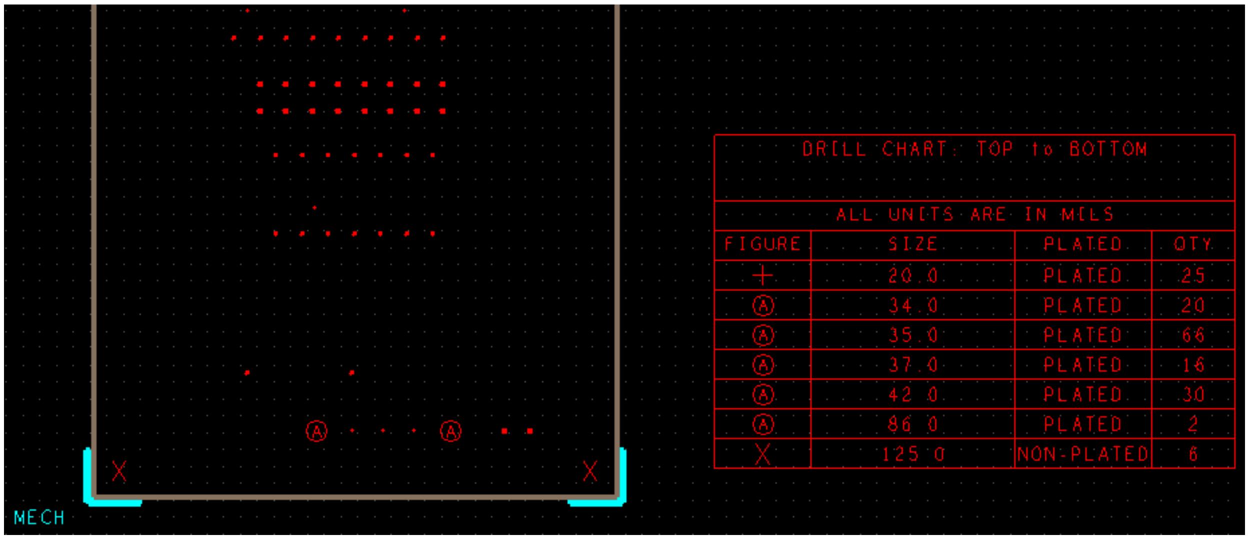
Total elements: 6

Press *OK* and place the new drill legend (or chart) as shown on the following page.

Visual drill targets appear on your design keyed to lined entries in the Drill Chart. Before discussing the chart and drill sizes, go ahead and first create a new film record and title it "MECH". For reference, be sure your new mechanical layer combines the following sub-classes; these should already be visible at this point.

MECH

MANUFACTURING / NCLEGEND-1-2	
BOARD GEOMETRY / OUTLINE	
BOARD GEOMETRY / CUT_MARKS	Add APC text to this layer



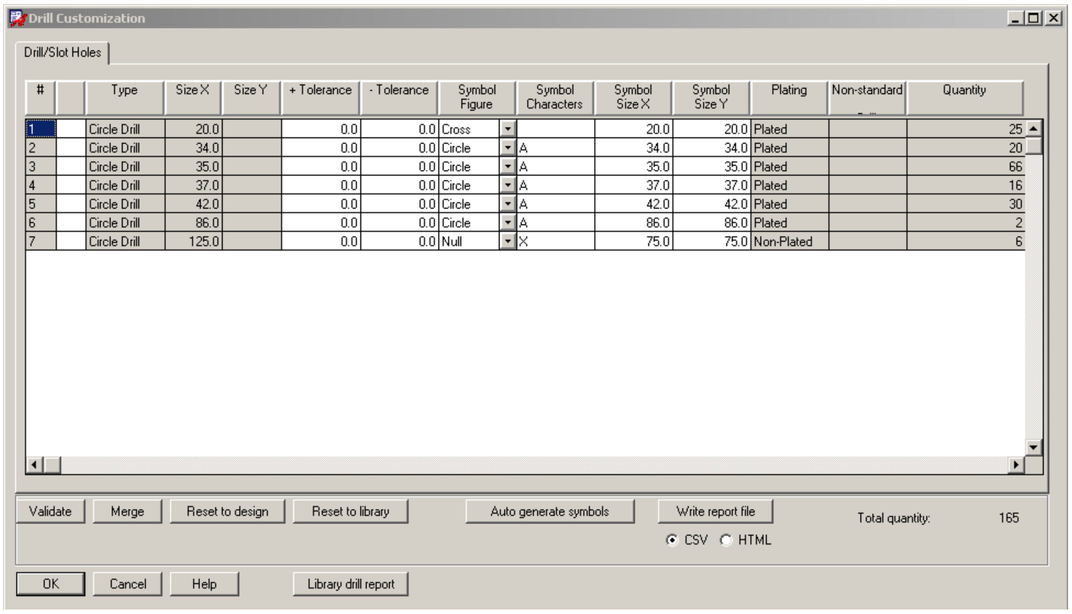
DRILL CHART: TOP to BOTTOM

ALL UNITS ARE IN MILS

FIGURE	SIZE	PLATED	QTY
+	20.0	PLATED	25
Ⓐ	34.0	PLATED	20
Ⓐ	35.0	PLATED	66
Ⓐ	37.0	PLATED	16
Ⓐ	42.0	PLATED	30
Ⓐ	86.0	PLATED	2
X	125.0	NON-PLATED	6

Checking Drill Sizes

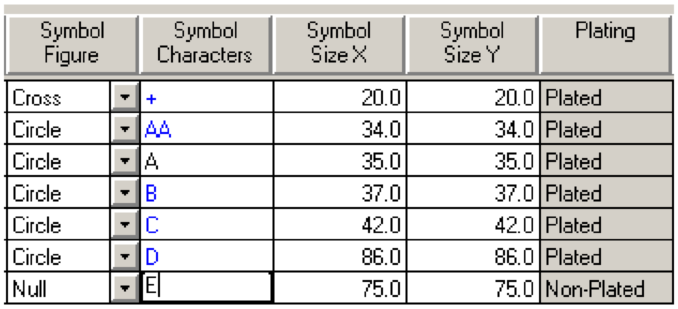
APC requires that we limit the drill rack to specific sizes, or they charge extra. Check drills by referring to the **Drill Chart** you just generated. Observe the chart associates symbols for each drill size keyed to every drill location. Determine strange drill sizes by inspecting the chart and locating where they are on the board using the symbols. Before changing any rogue drill be sure the new size will work mechanically. If you're uncertain whether to increase or decrease a particular drill, the safest direction is larger. Before trying to find and fix drill sizes (note the 34 and 37 mil drills shown above need changing to 35 and 38 respectively), we need to re-map the figure legend, since most drills are mapped to symbol "circle A". This can be done by individually editing each padstack in the *Padstack Designer*, or collectively using the *Drill Customization* spreadsheet; we'll use the spreadsheet since it's *much* easier. Access the spreadsheet by menu sequence **Manufacture ⟶ NC ⟶ Drill Customization…** **(**or use the the icon). The initial spreadsheet looks like the following:



#		Type	Size X	Size Y	+ Tolerance	- Tolerance	Symbol Figure	Symbol Characters	Symbol Size X	Symbol Size Y	Plating	Non-standard	Quantity
1		Circle Drill	20.0		0.0	0.0	Cross		20.0	20.0	Plated		25
2		Circle Drill	34.0		0.0	0.0	Circle	A	34.0	34.0	Plated		20
3		Circle Drill	35.0		0.0	0.0	Circle	A	35.0	35.0	Plated		66
4		Circle Drill	37.0		0.0	0.0	Circle	A	37.0	37.0	Plated		16
5		Circle Drill	42.0		0.0	0.0	Circle	A	42.0	42.0	Plated		30
6		Circle Drill	86.0		0.0	0.0	Circle	A	86.0	86.0	Plated		2
7		Circle Drill	125.0		0.0	0.0	Null	X	75.0	75.0	Non-Plated		6

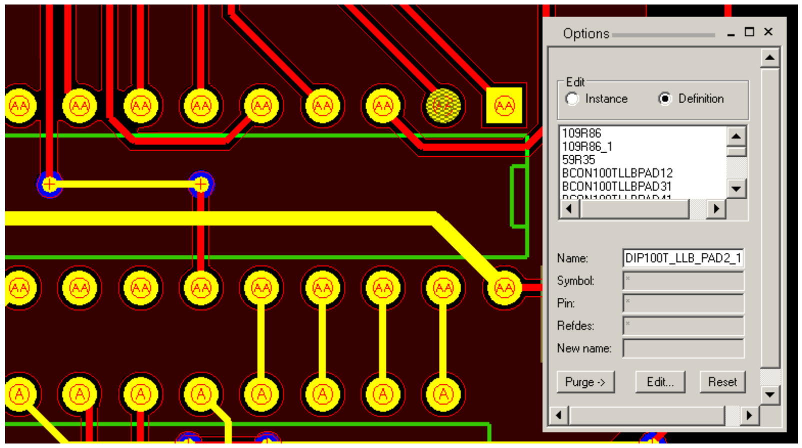
Total quantity: 165

This spreadsheet will display all drill parameters, but it can only be used to edit *symbols* , not actual drill sizes; that must be done only in the padstack design editor, which I'll show you after editing the spreadsheet. Edit the spreadsheet symbols as shown in the partial view below. Since I chose "circle A" for the 35 mil size, I chose "circle AA" to be a temporary notation to help find the 34 mil padstacks that will then be changed to "circle A" and 35mils. Also I chose "circle B" for the 38 mil size and will find and edit the 37 mil instances accordingly, making them 38mils.



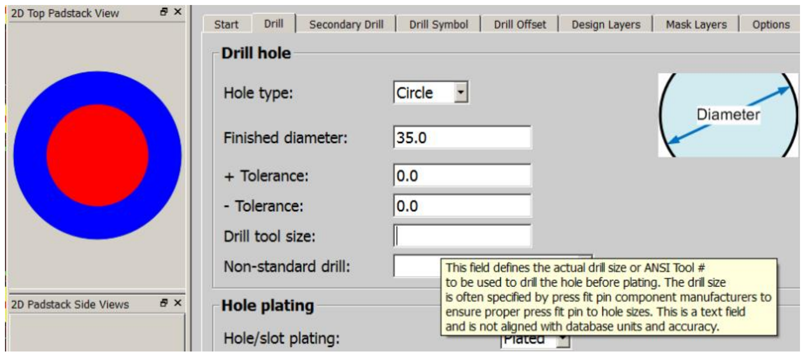
Symbol Figure	Symbol Characters	Symbol Size X	Symbol Size Y	Plating
Cross	+	20.0	20.0	Plated
Circle	AA	34.0	34.0	Plated
Circle	A	35.0	35.0	Plated
Circle	B	37.0	37.0	Plated
Circle	C	42.0	42.0	Plated
Circle	D	86.0	86.0	Plated
Null	E	75.0	75.0	Non-Plated

When you're done, press *OK*. The newly edited visual symbols will appear in your design, (but will only appear in the drill chart when it is recreated). This will enable us to find the 34 mil padstacks and change their drill sizes to 35 mils. Pan around, find the padtstacks and turn on the top copper sub-classes using the Visibility panel (I also turned bottom copper layer; this is optional). (To see the drill legend character symbols, the active class and subclass should be *Manufacturing / Nclegend-1*-2.) An example is shown below:



Name: DIP100T_LLB_PAD2_1

We notice the wrong 34 mil padstack, noted now as AA, is associated with the IC DIP packages. Observe these DIP footprints have two separate padstacks, one for pin-1 which is always square, and one for the rest which are round. Both of them must be changed individually. Choose **Tools** ⟶ **Padstack** ⟶ **Modify Design Padstack…** and select either of the large "circle AA" padstacks. Remember these identify the *drill size*, not the pad shape. Here, I selected the round padstack immediately to the left of the the square pad. Allegro will immediately identify which padstack it is in the *Name* field of the Options dialog, as shown it is padstack "DIP100T_LLB_PAD2_1. If you knew the particular padstack's name (which we don't here), it could also be selected directly from the pull-down list. Press the *Edit* button to open it for editing in the *Padstack Editor* [19] and select the *Drill* tab. The "Finished diameter" will show 34.0, as we expected. New to ver 17.2 is the "Drill tool size". Most likely blank, this is the field that should be edited for the *actual drill size*, while the finished diameter above it can then be adjusted to account for the *plating allowance*, the reduced resulting hole size once the via has been plated through. If the drill tool size is left blank, Cadence will default to using the finished diameter as the actual drill size. Why does this matter? If you tell a board house that your vias are finished sizes they will automatically increase the actual drill used by 3mils. For the board house we'll be using in this tutorial, actual drill sizes will be specified, not finished PTH sizes. So we'll leave the new field blank and put the drill size we want in the finished diameter field. Hence, enter the fields as shown in the partial screen shot. I left the mouse hovering over the new field so you could see the associated message.



Drill hole

Hole type: Circle

Finished diameter: 35.0

+ Tolerance: 0.0

- Tolerance: 0.0

Drill tool size:

Non-standard drill:

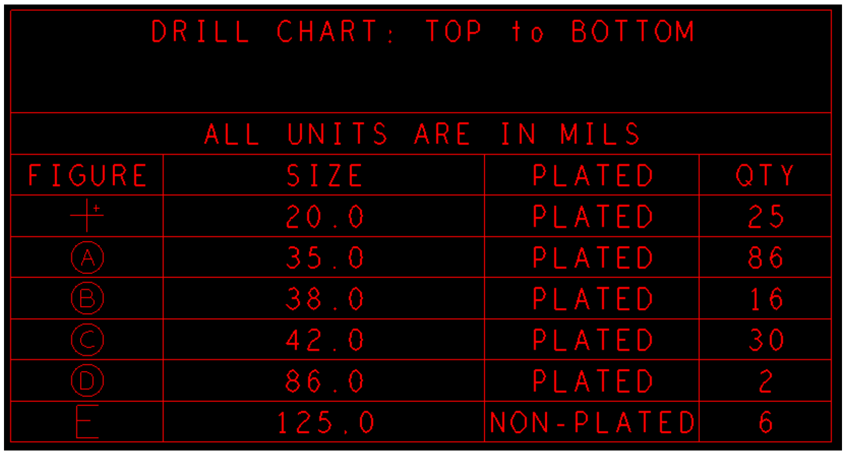
This field defines the actual drill size or ANSI Tool # to be used to drill the hole before plating. The drill size is often specified by press fit pin component manufacturers to ensure proper press fit pin to hole sizes. This is a text field and is not aligned with database units and accuracy.

Hole plating

Hole/slot plating: Plated

[19] See pg. 35 for a review of the Padstack Designer.

Note that you can also edit the drill symbol by selecting the *Drill Symbol* tab. Do that now and make the symbol 35mils to agree with the tool size and change the symbol character from “AA” to “A”. Save these changes back to the design by selecting menu sequence: **File ⟶ Update to Design and Exit**. Disregard the warnings in the *Padstack Errors* dialog (just close it; I’ll discuss these fields in the next lab) and select *Yes*. The editor should close. Proceed to repeat this process for the square “circle AA” padstacks. Similarly, find any 37mil padstacks (square or round), and edit them to use 38mil drills. It’s easiest to also increase the symbol size to 38 mils directly in the padstack editor, but this can also be done later in the Drill Customization spreadsheet. Note too that since we are editing *Definitions* and not *Instances*, editing usually affects a large number, which is what we want. Recreate the drill chart as you did earlier and the newly edited padstacks will be reflected in the chart and target symbols. This will also be reflected in the spreadsheet as well.



DRILL CHART: TOP to BOTTOM

ALL UNITS ARE IN MILS

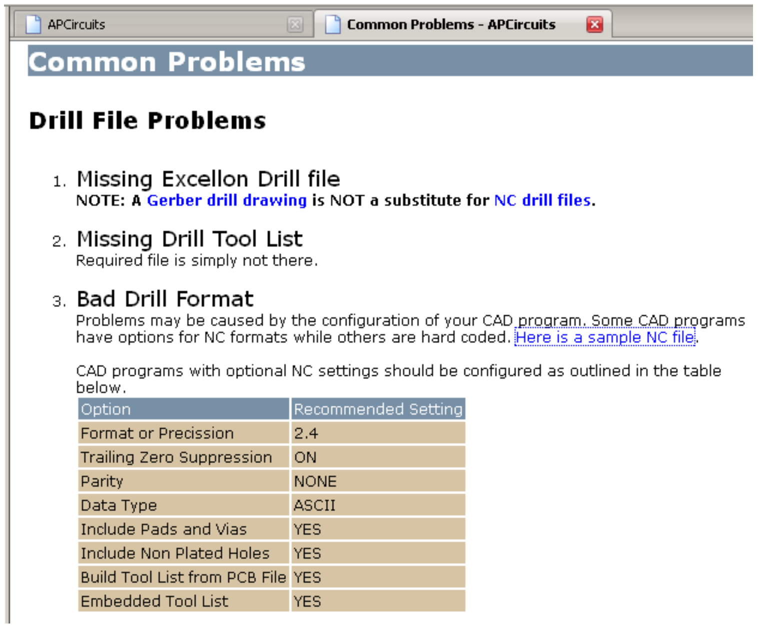
FIGURE	SIZE	PLATED	QTY
+	20.0	PLATED	25
Ⓐ	35.0	PLATED	86
Ⓑ	38.0	PLATED	16
Ⓒ	42.0	PLATED	30
Ⓓ	86.0	PLATED	2
E	125.0	NON-PLATED	6

Generally, when editing drill sizes, be sure to check and adjust the top and bottom pad diameters if necessary. Increasing a drill size by 1 mil means a corresponding *decrease* in the resulting annular ring by 0.5 mil; *keep the annular ring at least 10 mils*. This is done on the *Layers* tab of the *Padstack Designer* [20] by changing the padsize in the *Regular Pad* pad entry field as necessary. This wasn’t necessary for the edits we just did. I’ll show you how do this in lab 2.1.

[20] See pg. 36.

Drill Files

Besides the usual Gerber files, board houses also need special information describing what drill sizes to use and where to drill them. Always check with the board house you intend to submit your design to for details regarding what they want. We know in our case that APC wants two classic files: 1) *NC Drill file* and 2) *Drill List*. The two screen shots below are both from APC's website, showing examples of what they want regarding drill files. I have included on the first screen-shot only the first three points, of which the third is the most important dealing with the format.



Common Problems

Drill File Problems

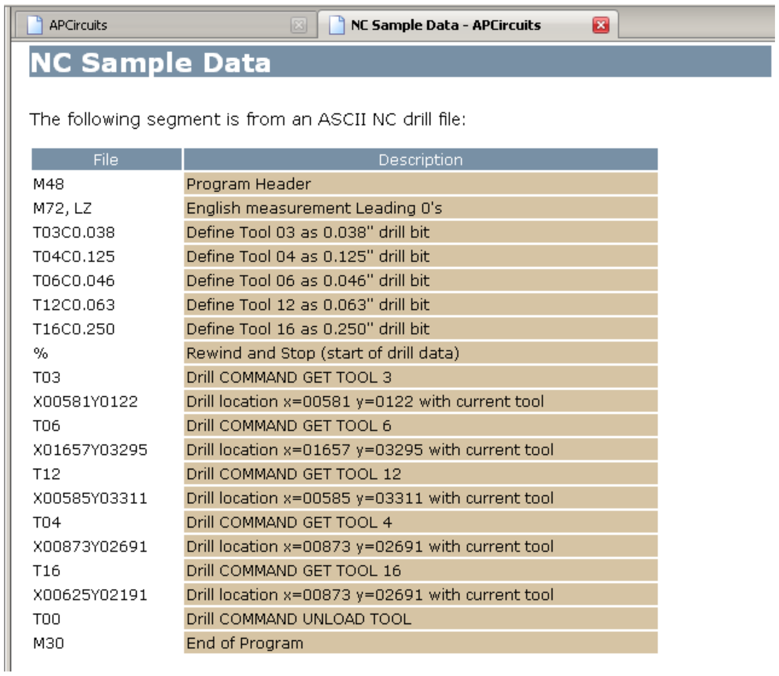
1. Missing Excellon Drill file
   NOTE: A Gerber drill drawing is NOT a substitute for NC drill files.
2. Missing Drill Tool List
   Required file is simply not there.
3. Bad Drill Format
   Problems may be caused by the configuration of your CAD program. Some CAD programs have options for NC formats while others are hard coded. Here is a sample NC file.

   CAD programs with optional NC settings should be configured as outlined in the table below.

Option	Recommended Setting
Format or Precission	2.4
Trailing Zero Suppression	ON
Parity	NONE
Data Type	ASCII
Include Pads and Vias	YES
Include Non Plated Holes	YES
Build Tool List from PCB File	YES
Embedded Tool List	YES

APC tells us they want two files:

- **Excellon Drill File**. This file contains an ASCII formatted map of all drills used and their physical X-Y positions.
- **Drill Tool List**. This ASCII file is a simple list of drill sizes associated with tool numbers used in the Excellon Drill file.



NC Sample Data

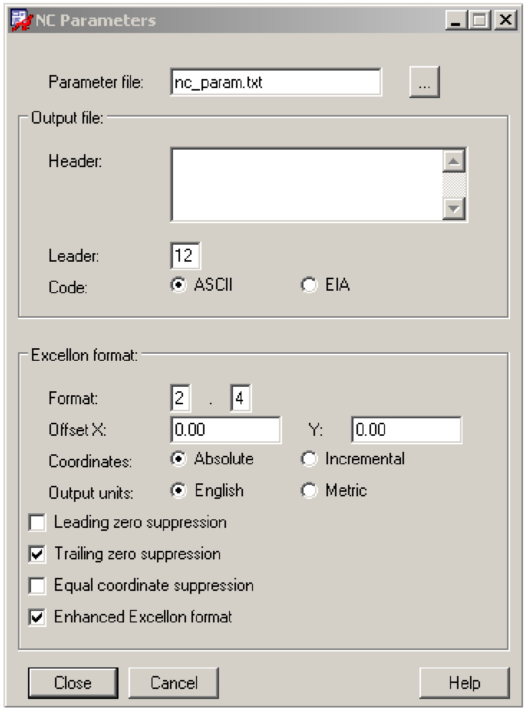
The following segment is from an ASCII NC drill file:

File	Description
M48	Program Header
M72, LZ	English measurement Leading 0's
T03C0.038	Define Tool 03 as 0.038" drill bit
T04C0.125	Define Tool 04 as 0.125" drill bit
T06C0.046	Define Tool 06 as 0.046" drill bit
T12C0.063	Define Tool 12 as 0.063" drill bit
T16C0.250	Define Tool 16 as 0.250" drill bit
%	Rewind and Stop (start of drill data)
T03	Drill COMMAND GET TOOL 3
X00581Y0122	Drill location x=00581 y=0122 with current tool
T06	Drill COMMAND GET TOOL 6
X01657Y03295	Drill location x=01657 y=03295 with current tool
T12	Drill COMMAND GET TOOL 12
X00585Y03311	Drill location x=00585 y=03311 with current tool
T04	Drill COMMAND GET TOOL 4
X00873Y02691	Drill location x=00873 y=02691 with current tool
T16	Drill COMMAND GET TOOL 16
X00625Y02191	Drill location x=00873 y=02691 with current tool
T00	Drill COMMAND UNLOAD TOOL
M30	End of Program

APC further shows us an example of what they want in a typical Excellon-formatted NC drill file, as shown left.

Based on the web information above, I set the parameters in the **NC Drill** and **NC Parameters** dialogs to experimentally produce ASCII output files that most agreed with the parameters wanted by APC.

First, specify the parameters by choosing **Manufacture → NC → NC Parameters…** or use the icon. Enter the parameters as shown in the *NC Parameters* dialog.


NC Parameters

Parameter file: nc_param.txt

Output file:

Header:

Leader: 12

Code: [✓ ASCII] EIA

Excellon format:

Format: 2 . 4

Offset X: 0.00 Y: 0.00

Coordinates: [✓ Absolute] Incremental

Output units: [✓ English] Metric

☐ Leading zero suppression

☑ Trailing zero suppression

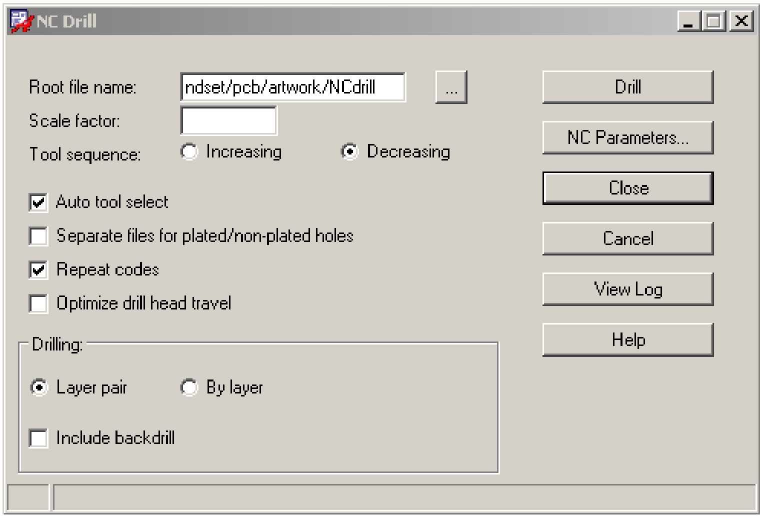
☐ Equal coordinate suppression

☑ Enhanced Excellon format

The “parameter” file *nc_param.txt* (this is *not* related to the *.prm files discussed at the beginning of this lab) saves the setup parameters shown here in a custom file specifically associated, in this case, with APC. This file is placed in your …Handset/pcb/artwork folder. After defining the parameters select *Close*. Allegro will write the file out to your artwork folder.

Be sure “Enhanced Excellon format is checked, or the drill list won’t automatically be included inside the NC drill file (the “nc_param.txt” parameter file). If this is inadvertently omitted, most board house will look inside your drill tool list. If you intend to output artwork for later inputting into CircuitCam here at UCSC, this options MUST be checked.

Setup and generate the two drill files as follows: Choose menu sequence **Manufacture → NC → NC Drill**. Fill in the *Root file name* as **NCdrill.** Then press *Drill*. Allegro will automatically generate two files:


NC Drill

Root file name: ndset/pcb/artwork/NCdrill

Scale factor:

Tool sequence: Increasing [✓ Decreasing]

☑ Auto tool select

☐ Separate files for plated/non-plated holes

☑ Repeat codes

☐ Optimize drill head travel

Drilling:

[✓ Layer pair] By layer

☐ Include backdrill

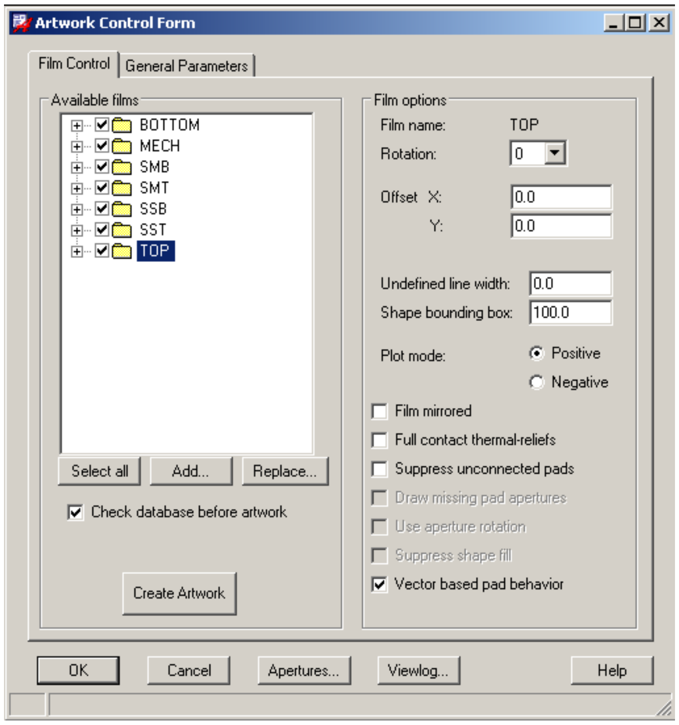
1. **NCdrill-1-2.drl**. This is the NC drill file. Added notation and extension are automatically added. “-1-2” refers to the layers affected, in this case layer 1 (top) through to layer 2 (bottom).

2. **nc_tools_auto.txt.** This is the drill tool list containing tool numbers mapped to drill diameters used in the NC drill file above.

Check to make sure both of these files have been successfully created and placed in your artwork folder. Note that Allegro creates backups of existing files in this folder by appending “,1” to the extension. All of these can be deleted once you’ve finished creating final versions.

Pulling it all together

After creating the two drill files, generate all the artwork files. Access the *Artwork Control Form*, check all the film records and press *Create Artwork.*



Available films: ☑ BOTTOM, ☑ MECH, ☑ SMB, ☑ SMT, ☑ SSB, ☑ SST, ☑ TOP

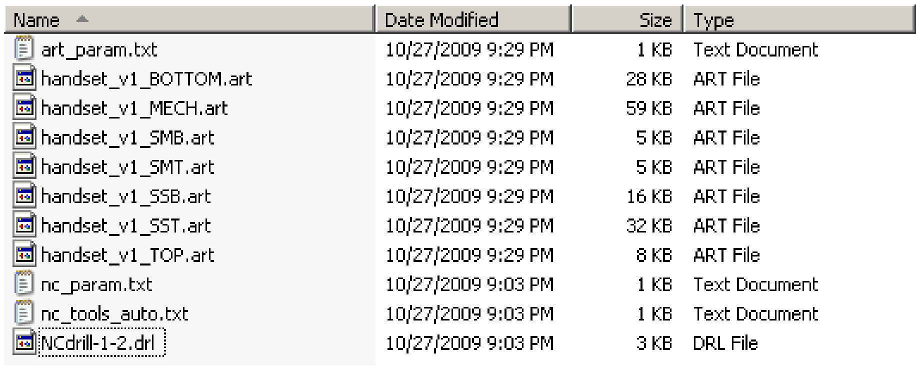
☑ Check database before artwork

Film options

Film name:	TOP
Rotation:	0
Offset X:	0.0
Y:	0.0
Undefined line width:	0.0
Shape bounding box:	100.0
Plot mode:	[✓ Positive] Negative

☐ Film mirrored
☐ Full contact thermal-reliefs
☐ Suppress unconnected pads
☐ Draw missing pad apertures
☐ Use aperture rotation
☐ Suppress shape fill
☑ Vector based pad behavior

The screen-shot below summarizes the files associated with the manufacturing phase you have just completed. Notice how relatively small these files are for the amount of work that went into creating this PCB!



Name	Date Modified	Size	Type
art_param.txt	10/27/2009 9:29 PM	1 KB	Text Document
handset_v1_BOTTOM.art	10/27/2009 9:29 PM	28 KB	ART File
handset_v1_MECH.art	10/27/2009 9:29 PM	59 KB	ART File
handset_v1_SMB.art	10/27/2009 9:29 PM	5 KB	ART File
handset_v1_SMT.art	10/27/2009 9:29 PM	5 KB	ART File
handset_v1_SSB.art	10/27/2009 9:29 PM	16 KB	ART File
handset_v1_SST.art	10/27/2009 9:29 PM	32 KB	ART File
handset_v1_TOP.art	10/27/2009 9:29 PM	8 KB	ART File
nc_param.txt	10/27/2009 9:03 PM	1 KB	Text Document
nc_tools_auto.txt	10/27/2009 9:03 PM	1 KB	Text Document
NCdrill-1-2.drl	10/27/2009 9:03 PM	3 KB	DRL File

Generally, before sending any of these files to any boardhouse, they would normally be verified using a third-party Gerber viewer. I will show you how to do this in lab-3. For now, we'll assume the Gerber and drill files have been verified and you're ready to send them off to APC! The summaries below list files minimally associated with each service type:

APC Basic Prototype Files - The following files minimal files are required by APC:

- handset_v1_TOP.art
- handset_v1_BOTTOM.art
- handset_v1_MECH
- nc_tool_auto.txt
- NCdrill-1-2.drl

APC Plus Prototype Files – Had you selected their Plus prototype service instead, the following files would then be submitted to APC:

- All the files noted above for the Basic Prototype service.
- handset_v1_SMT.art
- handset_v1_SMB.art
- handset_v1_SST.art (optional depending on silk-screen options).
- handset_v1_SSB.art (optional depending on silk-screen options).

In the past, APC accepted these design files zipped and sent via ftp or as an email attachement, along with a single page submission form with relevant customer information; it is still available on the website. In about 2012, they developed a convenient Javascript submission form that should be be used instead to submit these files. They prefer this, since it combines the customer information and definitively maps artwork filenames with their intended function (top, bottom etc.) and doesn't rely soley on filenames; therefore it is much less error-prone. Download it from their website and review its features. After filling it out, the form will submit your design via ftp. (Don't actually submit it, however!).

5. Conclusion

This concludes the second laboratory. Laboratory 2.1 will build on this one and cover specifically how to make footprints and padstacks.

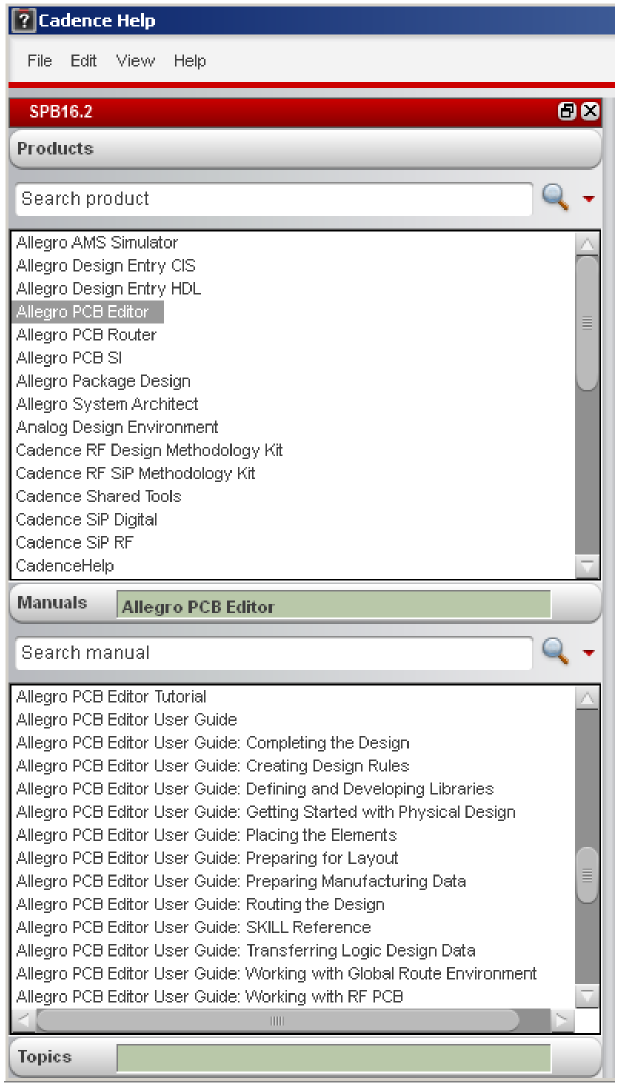
Oral Evaluation of your work for grading:

Your grade for this laboratory will be based on showing the instructor competence using Allegro and understanding things presented and discussed in the lab. Your understanding will be gauged by responses to any of the following topics and questions. You will be asked to answer or expound on three of them chosen randomly.

1. Demonstrate competence with the **Cline** and **Cline Segs** filters.
2. Demonstrate and explain how to work with **Text**.
3. Compare and constrast the three types of shapes used in this design.
4. How does Allegro implement classic traditional layers?
5. What are the basic layer types?
6. Explain how the two grids are used.
7. What is a constraint technology file? How is it used?
8. What is a parameter file? How is it used?
9. What is a void?
10. Padstacks. So what are they anyway?
11. Explain what's done during the **Manufacturing Phase**.
12. How are artwork, film and Gerber files related.
13. Use of the DRC.
14. Electrical and non-electrical components.
15. Define what is meant by "Standards of good engineering practice".
16. Why are wider track widths preferred for routing?
17. What is meant by "tacking"? Demonstrate how this is done.
18. How do you change the default via while routing?
19. How do you replace a single via?
20. What is meant by "Manhatten routing". When would you use this?
21. The **Status** report dialog. What's on it and why consult it? How is it used?
22. What's the difference between *copper pours* and *copper areas*? What does Cadence specifically call them?
23. When should *anti-copper areas* be used? What does Cadence call them?
24. Demonstrate how to change a drill size and what considerations need to be taken into account when doing this.
25. Explain the difference between a "draw" and a "flash".
26. What is meant by "annular ring" size and "plating allowance" with respect to vias? When do we need to consider them?
27. Solder reliefs. What are they? How are they used?
28. Compare solder reliefs with solder thieves.
29. What is meant by *copper balance?*
30. Discuss the difference between File views and Film views.
31. Show how to create a custom colorview.
32. What are the basic design complexity levels and associated spacings?
33. Compare and constrast RS274D with RS274X pcb export standards.
34. Cutting marks.
35. Islands.

APPENDIX-A. FINDING HELP RESOURCES

As I have already discussed earlier[21], the best place to find useful help resources is from within the Documentation Hierarchy engine. Doing this and selecting "product" *Allegro PCB Editor* brings up a long list of "manuals"as shown here. I have re-positioned the lower Manuals' window to bracket all of the relevant references in the Allegro PCB Editor User Guide. These are actually all chapters referenced from within the corresponding HTML reference, but accessible here as single, and therefore more useful, PDF files. I will reference these as we go along, indicating where certain topics can be found. You should always remember that several different front-end tool flows can be used ahead of Allegro, so keep in mind these manuals often treat the Design HDL and Design CIS front-end tools in the same document; some of the material won't be applicable and you'll need to carefully sift this out so as not to be confused.



Allegro AMS Simulator
Allegro Design Entry CIS
Allegro Design Entry HDL
Allegro PCB Editor
Allegro PCB Router
Allegro PCB SI
Allegro Package Design
Allegro System Architect
Analog Design Environment
Cadence RF Design Methodology Kit
Cadence RF SiP Methodology Kit
Cadence Shared Tools
Cadence SiP Digital
Cadence SiP RF
CadenceHelp

Manuals: Allegro PCB Editor

Allegro PCB Editor Tutorial
Allegro PCB Editor User Guide
Allegro PCB Editor User Guide: Completing the Design
Allegro PCB Editor User Guide: Creating Design Rules
Allegro PCB Editor User Guide: Defining and Developing Libraries
Allegro PCB Editor User Guide: Getting Started with Physical Design
Allegro PCB Editor User Guide: Placing the Elements
Allegro PCB Editor User Guide: Preparing for Layout
Allegro PCB Editor User Guide: Preparing Manufacturing Data
Allegro PCB Editor User Guide: Routing the Design
Allegro PCB Editor User Guide: SKILL Reference
Allegro PCB Editor User Guide: Transferring Logic Design Data
Allegro PCB Editor User Guide: Working with Global Route Environment
Allegro PCB Editor User Guide: Working with RF PCB

[21] See the Tip on page 42, Laboratory 2, Capture, for an in-depth introduction.

APPENDIX-B. FORWARD AND BACK ANNOTATION

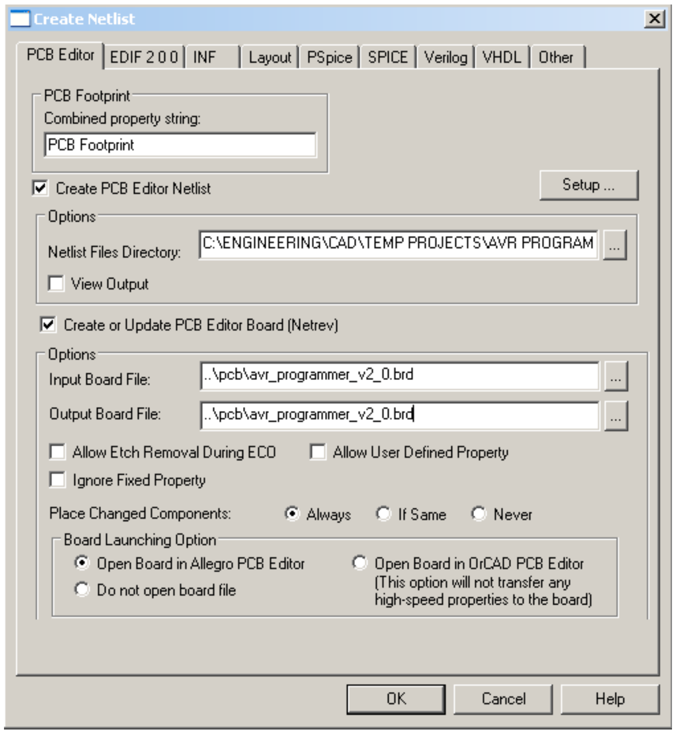
FORWARD ANNOTATION FROM CAPTURE TO ALLEGRO PCB - The objective is to reflect in-place changes to the current PCB design directly from Capture. This is tricky, since it's quite easy to erase your present *.brd file.

I'll discuss several conservative safe ways to do this. Suppose we're working with two design files:

The first is our schematic file, ***avr_programmer_v2.0.dsn*** in Capture.
The second is the PCB file, ***avr_programmer_v2.0.brd*** in Allegro.

Begin by saving the current board file as some other name, say ***avr_programmer_v2.0_backup.brd*** . This file is only useful as a fall-back in case we make a mistake in the following steps (never happens, right?). Then rebuild the netlist in Capture with the new changes, but define the input to be the current file and output board file to be a new one with an appropriate revision name …v2.0.1.brd (or browse to get the file location and name and edit the output file with the added revision).

***Caution*!** This action always generates a new *.brd file, and it will be empty if no input file was chosen. Therefore, creating a new temporary file is good practice, since after reviewing it, you can save it back under the original. In any case, if the output file is correct it will <u>absolutely be overwritten</u> with *something* – this is the reason for step one above. If you mess things up, reload the backup and immediately save it as the correct working version and try again.


Create Netlist
PCB Editor | EDIF 2 0 0 | INF | Layout | PSpice | SPICE | Verilog | VHDL | Other
Setup ...
OK
Cancel
Help

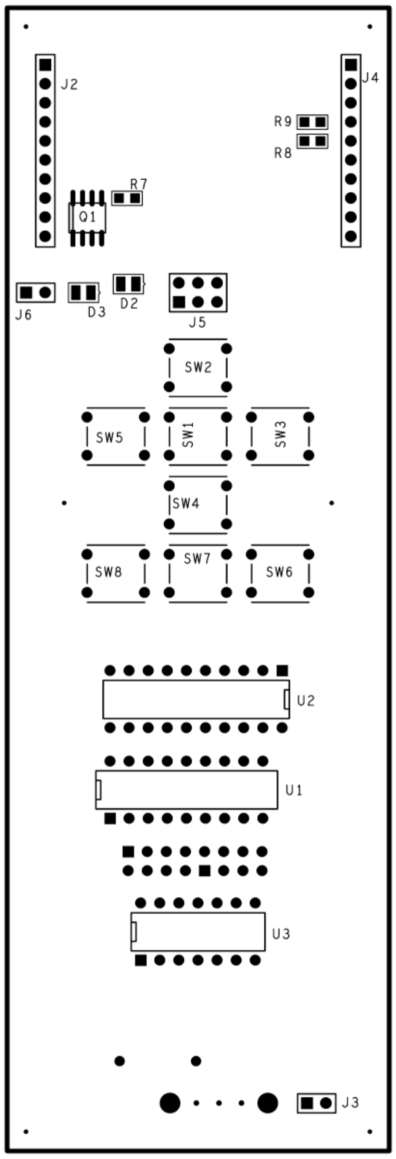
PCB Footprint
Combined property string: PCB Footprint
[✓] Create PCB Editor Netlist
Options
Netlist Files Directory: C:\ENGINEERING\CAD\TEMP PROJECTS\AVR PROGRAM
[ ] View Output
[✓] Create or Update PCB Editor Board (Netrev)
Options
Input Board File: ..\pcb\avr_programmer_v2_0.brd
Output Board File: ..\pcb\avr_programmer_v2_0.brd
[ ] Allow Etch Removal During ECO
[ ] Allow User Defined Property
[ ] Ignore Fixed Property
Place Changed Components: [✓ Always] If Same Never
Board Launching Option
[✓ Open Board in Allegro PCB Editor] Open Board in OrCAD PCB Editor (This option will not transfer any high-speed properties to the board)
Do not open board file

Another way is to simply *not* check the *Create or Update PCB Editor Board (Netrev)*. Run the new netlist and then manually re-import the logic into your working board file in Allegro. Be sure to make a backup first, however!

APPENDIX-C. PCB LAYOUT VIEWS

The following figures are meant to aid you in placing components and confirming how traces and various copper pours appear.

Laying out a PCB is generally constrained by the board's "form-factor", or physical dimensions and what it's overall purpose is. This PCB was designed to be used by High School students as a hand-held remote robotic control unit. Hence, placement of parts and traces were carefully chosen to minimized tactile interaction. Where possible, these were placed on the bottom of the board. This is most noticeable with the area about the switches.



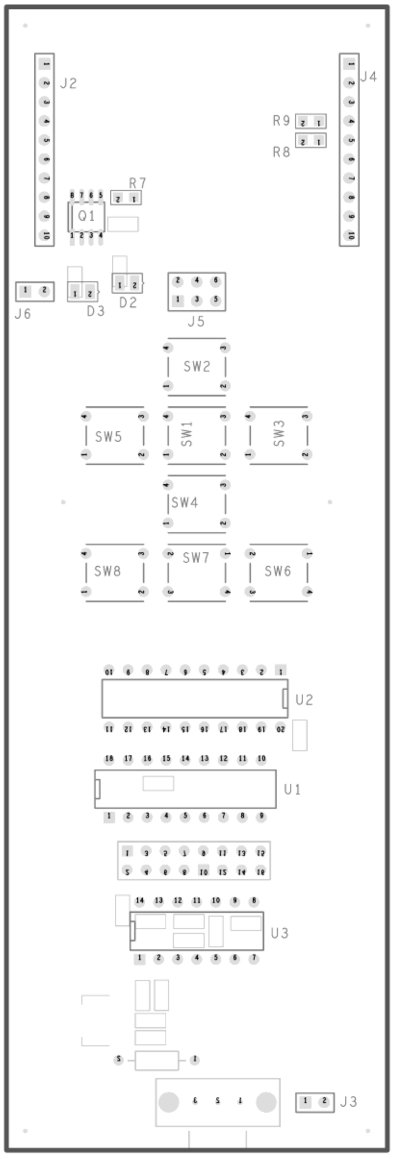


Fig. 1 (left). View showing the top-side of the board with reference designators for parts mounted only on this side. Note that parts having through-holes vias but no reference designator or outline *are mounted on the bottom-side*; only their pads will show on this side of the board.

Fig. 2 (right). View showing pin numbers along with the silkcreen bottom.

Gray Scale Legend:

Dark gray: SST (silkscreen top); these will also have reference designators.

Light gray: SSB(silkscreen bottom) showing part outlines on the bottom-side but without reference designators.

Note: how to make pin numbers visible is discussed in sec. 3.1, Lab 2 and briefly described here on the next page.

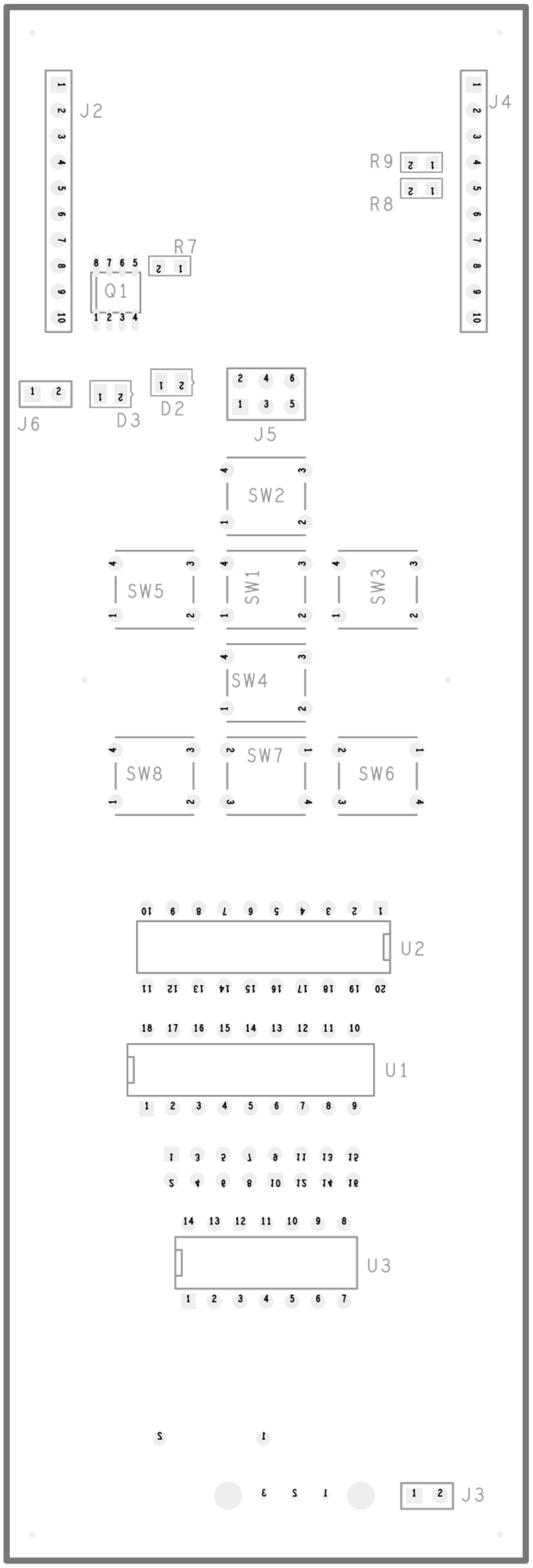
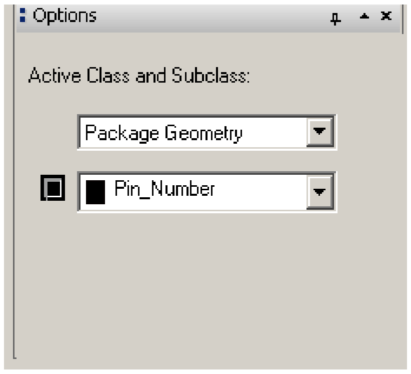


Fig. 3. Larger view of the pin numbers showing only SST (silkscreen top) and AST (assembly top) layers.

Note that pin numbers can me made visible at anytime by first selecting the the *Pin_Number* subclass found in the *Package Geometry* class and checking the small square checkbox immediately to the left of the subclass pull-down listbox; this is shown below (see sec 3.1, Lab2):


Options

Active Class and Subclass:

Package Geometry

Pin_Number

For this example, I changed the color of this subclass to black for higher contrast. Normally, toggling the small checkbox will display the current subclass color when visible and black when invisible.

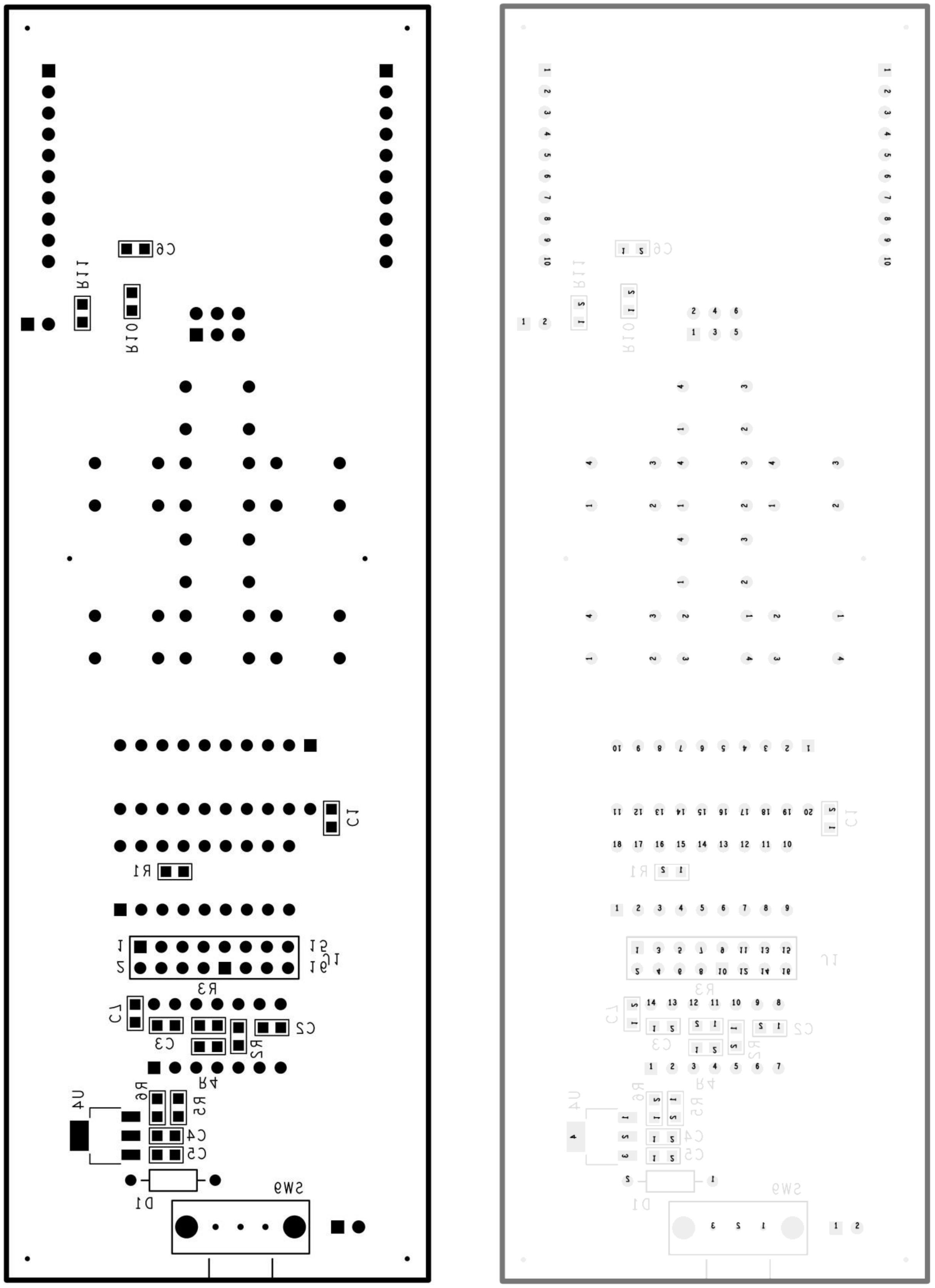


Fig. 4. Large side-by-side views of the bottom side with SSB and part outlines visible. Note that J2 has pin numbers labeled and visible too (left only). This was done to point out that even even though the part was mirrored to the bottom to orient the pin numbers for easy routing, the actual part was still mounted on the top-side.

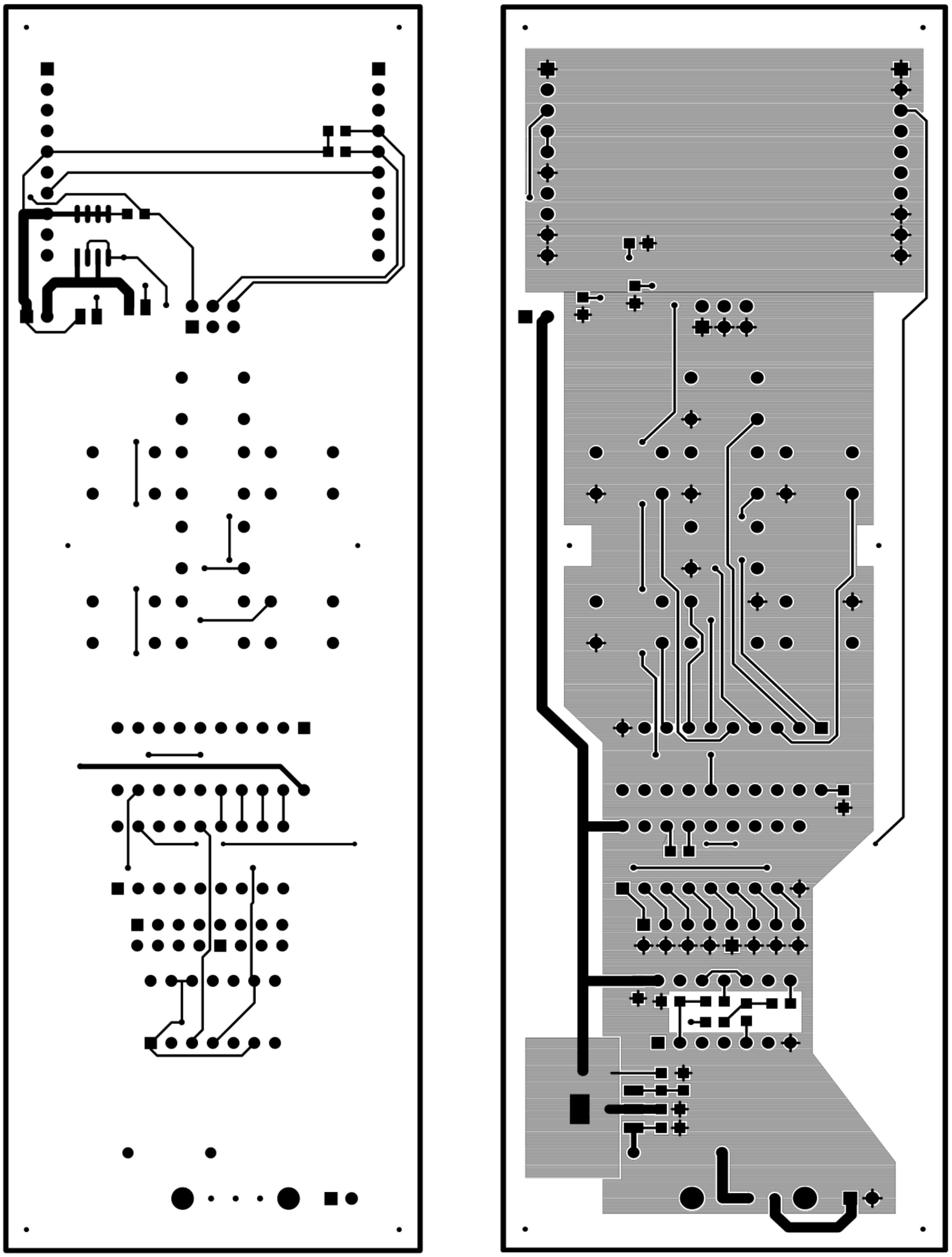


Fig. 5. Large side-by-side views of the top (left) and bottom (right) copper (etch) only. Note the large copper pour on the bottom has been greyed to distinguish it from the routed traces and via pads. When the board is actually made this would appear as a single sheet of copper other indistinguishable from the pads and traces.

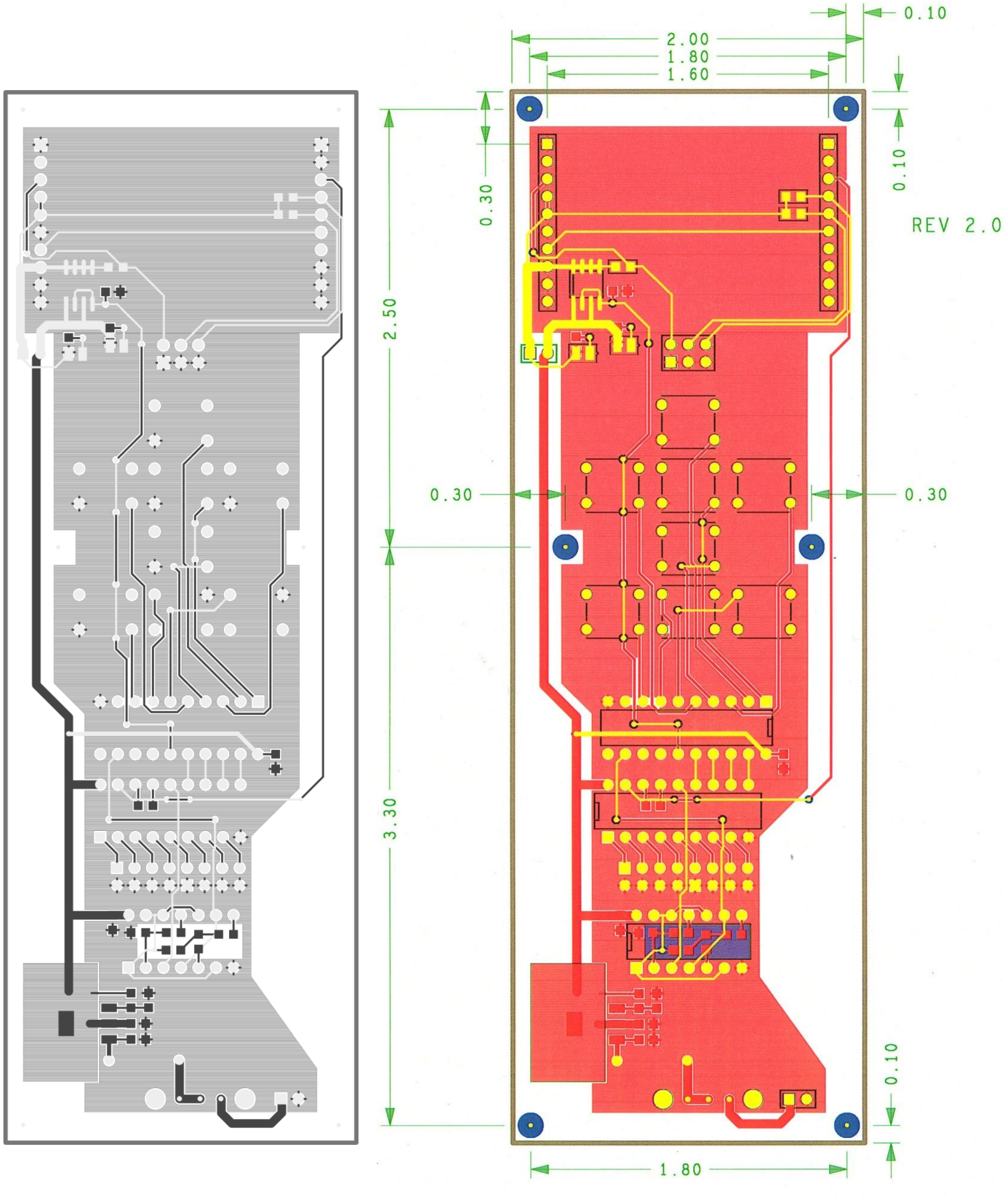


Fig. 6. Top and bottom copper layers (etch) in color and black and white.

Index