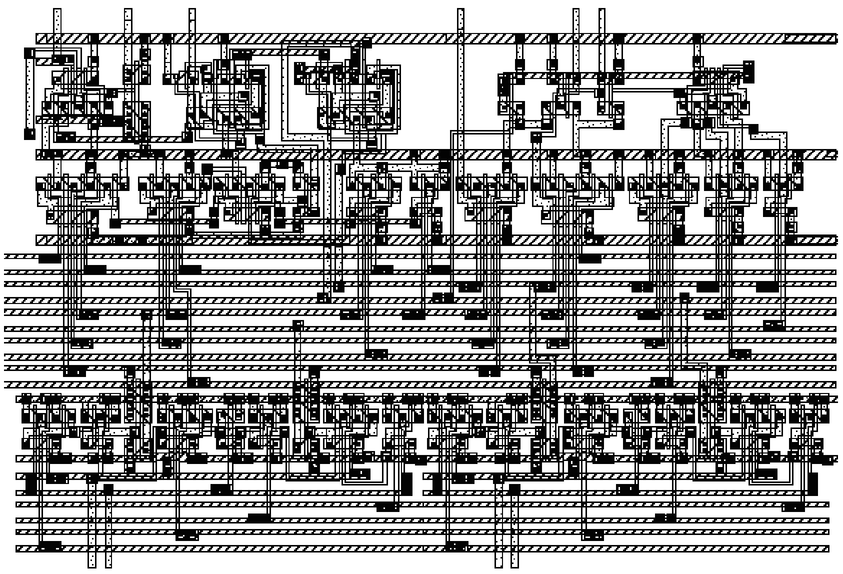
ICED™
Layout Editor for Windows

Classroom Tutorials



Version 4.xx

IC Editors, Inc.

Acknowledgments:

The majority of this manual was written or revised by Ference Professional Services in Sonora, CA. We are also responsible for formatting the text and creating the screen captures that illustrate the examples.

Michael Gentry of MGC, Inc. created the layout that is used on the cover and as a frontispiece. It is a section of a CMOS simulation of a 74181 4-bit ALU.

Table of Contents

Introduction

The tutorials in this manual cover all features of the ICED™ layout editor, a full-featured editor for integrated circuit mask sets. By the time you have completed all of the exercises you will be familiar with virtually every aspect of the editor and its associated tools.

This manual is designed for users who prefer to learn by doing. It takes a hands-on approach that allows you to see how the editor works by performing real tasks.

The Layout Editor Reference Manual covers all editor commands in more detail, but all of the more common uses of each command will be explained in the lessons and you will master their use by accomplishing typical layout tasks. You will be creating and modifying geometry, importing and exporting data, plotting, and even performing design rule verification.

In addition, you will learn about the importance of command file programming. You will create several useful command files that extend the capabilities of the editor by automating tasks and manipulating data with functions and other advanced programming features. This information is greatly expanded in the Command File Programmer's Reference Manual, available separately.

The tutorials are designed to be covered in the order provided. Tutorials closer to the end of the book assume a familiarity with the material earlier in the book. However, care was taken to make each tutorial relatively independent of the others. If you are already an experienced ICED™ user, you should be able to skip ahead to the more advanced tutorials if you prefer. Still, we are sure that even the most experienced users will find surprising features in the earlier lessons that will make it worth their time to read all of the lessons. Familiarity with helpful features can make time-consuming tasks trivially easy. Also, large amounts of time can be saved when common mistakes are avoided.

The **Layout Editor Basics Tutorial** is intended primarily for new users. It covers most of the material in the old User's Guide along with a few new lessons. The emphasis is on the basics of creation and modification of geometry.

Customizing ICED™ for Specific Projects will teach you the correct way to set up projects before opening the editor. Lessons will teach you how to create, protect, and share cell libraries. Other lessons cover the creation of technology-specific startup command files that control layer definitions and other environment settings stored in new cell files. We strongly recommend that all ICED™ users become familiar with this information.

The next tutorial, **More on Entering Commands**, covers features that allow you flexibility in how you execute commands. Menus and typed command syntax are covered here in more detail than in the basic tutorial. In addition, keyboard shortcuts and command files are introduced.

All commands that perform operations on existing geometry require that the appropriate component(s) are selected. Methods of selecting entire components, selecting only certain edges or vertices, and preventing certain components or layers from changes are covered in the **Selecting Components** tutorial. Once you have mastered these exercises, you can save large amounts of time in common tasks, since selecting the correct components is often the most time-consuming part of a modification operation.

The **More on Creating and Modifying Components** tutorial begins with lessons that teach you all about layer definition. It continues with exercises that create every type of ICED™ component, including text components, wires, cells, and arrays. All relevant modification methods are also covered including cutting, mirroring, rotating, aligning, and swapping components.

If you use components with sides not at 90º, you should perform the exercises in **Creating Components at Arbitrary Angles**. The creation of circular components, and the modification of shapes with skewed sides is included.

Methods of investigating and modifying nested cells are covered in **Exploring Cell Hierarchy**.

Manipulating the Display covers a variety of methods to change the way data is displayed in the view window. This includes more details on layer appearance such as color and pattern definition. Turning off the display of individual components or entire layers is also covered. Other methods that simplify the display of dense designs are included along with the use of visual aids such as display grids and wiring guides.

Methods of getting useful illustrations of your layout from your printer or plotting device are covered in **Plotting Layout Data**. This includes exercises on the more recent features of the editor that support bitmap export and plots with more than eight colors.

You will create ICED™ cells from sample Stream files and create Stream files from cell data in the lessons in **Importing and Exporting Data with GDSII-Stream Files**. Even if you do not routinely use this industry standard exchange format, you may find the lesson on scaling the size of a design during import, or the lesson on archive backups, relevant to your needs.

Command files are text files containing ICED™ commands. In the **Creating Useful Command Files** tutorial, you will create commands files that create and modify various types of geometry by manipulating coordinates and components. You will even create a command file that loops through all subcells of a design, snapping all coordinates to the resolution grid. Once you discover how easy it is to write and execute command files with these lessons, you will probably use them more frequently in your work, saving countless hours of tedium.

If you use the DRC program (the Design Rules Checker, available separately from IC Editors), you will find the lessons in **Using the Internal DRC** very valuable. You will perform Boolean operations on selected shapes and entire layers with ease. You will also execute a simple design rules set on a design and step through the errors found without ever leaving the layout editor.

The final tutorial includes a few short exercises that show you how to recover from various sorts of disasters. **Recovering from Mistakes or Crashes** includes information on cell backups and the journal file used to restore cells to a state they were in before a system crash or editing mistake.

Enjoy.

Layout Editor Basics Tutorial

This brief tutorial is intended to acquaint you with the way the ICED™ layout editor operates. It uses the most basic options of the most basic commands to demonstrate the general methods used to create and modify components. The tutorial focuses on using the menus supplied with ICED™ rather than on typing commands at the prompt or using command files. These topics will be covered in more detail in later tutorials.

Starting ICED™

All ICED™ products are designed to be launched from a DOS window. The best way to open a DOS window for this purpose is to use the ICED™ icon. This icon was created on your desktop during installation and displays the representation of a silicon wafer. Using the ICED™ icon sets the system's search path. This ensures that operating system looks in the correct directory for the program executable files. It also sets the current directory to the ICED™ directory, Q:\ICWIN[1]. Double click on the ICED icon now.

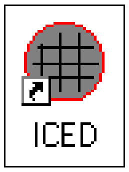


Figure 1: ICED desktop icon

ICED™ creates and stores data in cells. Each cell is stored as a separate file. We will use the TUTOR subdirectory to store the cell files created during this tutorial. You generally change to the directory where cell files are stored before launching the layout editor. To change the current directory, type the following command at the DOS prompt in the console window:

CD TUTOR <Enter>

If you have already performed any ICED™-related tutorials, there may be files in this directory left over from a previous tutorial. If this is not the case, or if you want to keep these files, you can omit the next step. However, it is best to start with an empty directory. Delete all files in the TUTOR directory by typing:

DEL *.* <Enter>

[1] Throughout this manual, Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. If you have installed the software on your C drive in the directory \ICED, you should replace Q: with C: and \ICWIN with \ICED.

ICED™ is usually launched by executing a project batch file that sets several environment variables and ICED™ command line options. The installation provides you with a sample batch file for this tutorial, Q:\ICWIN[2]\ICWIN.BAT. If you use this batch file, all you have to supply is the name of a cell to begin editing.

We will use this batch file to create a new cell with the name "MYCELL". At the DOS prompt, type:

ICWIN MYCELL <Enter>

When ICED™ starts up, it checks to see if the cell you have specified already exists or if it must be created. For new cells, ICED™ executes the startup command file specified in the command line in the batch file. (A listing of this startup command file is provided later on page 48.) This startup command file contains commands that define layer names and properties along with other parameters. After the startup command file is complete, the editor window will be displayed.

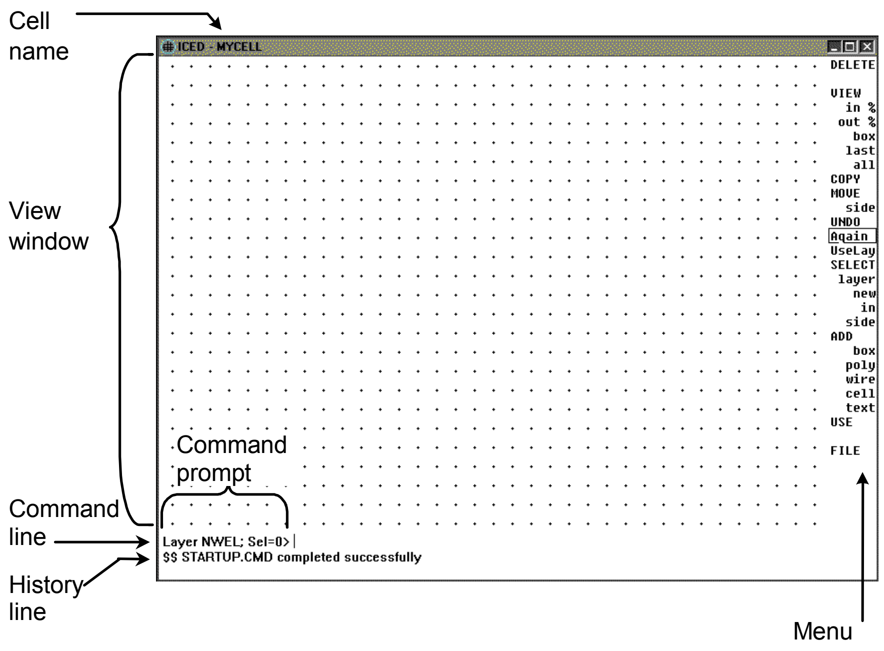
The ICED™ Window

After executing the startup command file, ICED™ displays a view window, a menu on the right side of the window, and the command and history lines on the bottom of the window. See Figure 2. An echo of the last command executed from the startup command file will be present on the history line.

The menu cursor indicates the current menu item with a box. In this case, the selected item is **Again**.

Type any letter on the keyboard repeatedly and you will see that you are adding characters at the command line cursor on the command line. The area on the same line to the left of the '>' symbol is called the command prompt. It displays two pieces of information, the current default layer and the number of selected components. Clear the typing on the command line now by pressing both mouse buttons at the same time.

[2] Remember that Q: and \ICWIN are used throughout the manual to represent the drive and directory where you have installed the ICED™ software.



```
Layer NWEL; Sel=0> |
$$ STARTUP.CMD completed successfully
```

Figure 2: Elements of the ICED™ window.

Closing the Layout Editor

If you want to interrupt this tutorial you can do so by following the instructions in Closing the Editor Normally on page 30. If you are using the free demo version, follow the instructions in Journal Files and Recovery from System Failure on page 29.

Entering Commands

Commands may be executed with any of the following methods:

- Selecting them from the menu using the mouse,
- Typing them on the command line using the keyboard, or
- Executing command files.

In this tutorial, most commands will be selected from the menus using the mouse.

Mouse Functions - An Overview

ICED™ works with either a two-button or a three-button mouse. The functions associated with each button change depending on the operation currently being executed.

In general, the mouse can only be used to select menu items until you finish selecting a command. Once you are executing a command that requires coordinate information, the mouse can only be used to digitize coordinates. Once the command is complete, the mouse can again be used to navigate through the menus.

The more common uses of the mouse are summarized in the simplified tables in Figure 3. We will expand this table with more detail in a later tutorial on page 83.

While a menu is displayed	
Left button	Selects menu entries
Right button	Cycles through top-level menus Indicates that you are finished selecting items from a menu list
Both buttons simultaneously	Cancels current menu selection and returns to first top-level menu
Center button	Ignored

While digitizing coordinates	
Left button	Digitizes coordinates
Right button	Indicates that you have finished digitizing coordinates of a wire or polygon
Both buttons simultaneously	Cancels the current command
Center button or <Esc> key	Invokes the nested view menu

Figure 3: Main uses of the mouse buttons

Using the Menus

ICED™ provides you with three top-level menus. Press the right mouse button once to see the second top-level menu. Press it again to see the third menu, and a third time to return to the first top-level menu. Now press the space bar on the keyboard. It also rotates the top-level menus. You can return to the first menu from anyplace in the menu system by pressing the left and right mouse buttons simultaneously. Press them now.

Notice that whenever you return to the first top-level menu, the menu cursor is positioned over the **Again** entry. The menu system generally initializes the cursor position to the same entry every time a given menu is displayed.

Some items on the menu open sub-menus when you select them. For example the ADD item in the first top-level menu has many options. Move the menu cursor until the word "ADD" is selected on the first top-level menu, then click the left mouse button. You can see a list of component types to choose from. One of these component types is "BOX".

We can describe this menu selection with the following shorthand:

1:**ADD → BOX**

The number '1' indicates the number of the top-level menu. In this case the item we want is on the first top-level menu. "ADD" is the item you choose by clicking the left mouse button while the item is selected by the menu cursor. After making this selection, you left-click the item "BOX" on the sub-menu.

Press both mouse buttons now to return to the first top-level menu. Note that "box" is also available right on the first top-level menu under the "ADD" item heading. Both "BOX" menu items access the same ICED™ command, "ADD BOX". The item is repeated on the first top-level menu under the "ADD" heading for convenience since it is used frequently.

We can describe this menu selection with the following shorthand:

1:(ADD)**box**

In this case, "ADD" is only a heading under which you locate the item to select. The item you need to select is in bold type, while the heading will be in normal type and enclosed by parentheses.

We will use this syntax for the rest of the tutorials.

Adding a Box

Using the left mouse button, select the ADD BOX command by clicking on:

1:(ADD)**box**

Now move the cursor around the screen with the mouse. As you move the mouse, the current coordinates of the cursor are displayed below the history line at the bottom of the window. Digitize the first corner of the box by pressing and releasing the left mouse button. Do not hold the mouse button down while you drag the mouse. Now move the cursor to another location on the screen and press the left mouse button again. The box has now been added to the drawing on the default layer. (We'll see how to change the default layer on the next page.)

Changing the View Window

You can use a variety of commands to change the view window. In addition, you can use the arrow keys to pan the view window. Holding down the <Ctrl> key while pressing an arrow key will cause the view window to shift in that direction. Hold down the <Ctrl> key and press the < ↑ > key, then the < ↓ > key. (If commands appear on the command line instead, press the <Esc> key to clear the command area and try the arrow keys again without pressing the <Ctrl> key. If you are using the numeric keypad arrow keys, make sure your keyboard's NumLock light is off.)

The VIEW ALL command changes the view window so that everything in the drawing is displayed. Click:

1:(VIEW)**all**

You can also change the size of the view window by using the zoom commands VIEW IN and VIEW OUT. To zoom out by a factor of 2, click the following options:

1:(VIEW)**out % → 2.0**

Setting the Default Layer

The default layer is the layer to which components are added, unless you override the layer specifically. Use the following menu items now to change the current default layer.

1:**UseLay → M1**

Until you change the default layer to another layer, any component added with the ADD menu options will be added to layer M1.

The UseLay operation continues at this point with an ADD menu allowing you to add a component on the new default layer. (In cases where you do not want to add a component immediately, you can simply return to the main menu by clicking the **MAIN** option, but don't do this now.)

Adding a Polygon

After you use UseLay to choose a default layer, a short ADD menu is automatically displayed. Select **POLY** from the menu. The menu will disappear. The program is now waiting for you to digitize the points of a polygon.

Move the cursor to the upper right area of the window and press the left mouse button. Notice that the first point of the polygon still has an X marking it as the starting point of the polygon. Continue to select polygon vertices.

Notice that if you draw a line that doubles back on itself, the doubled over section will disappear. Thus, if you draw a side that is too long you can correct it by doubling back and redigitizing the vertex in the correct position. If the side is too short you can extend it. This does *not* add extra vertices to a polygon.

Finish the polygon by adding vertices until it is closed. The best way to tell ICED™ that you have completed the polygon is to redigitize the starting vertex. Be sure to align the cursor carefully over the X marking the starting vertex. You can also tell ICED™ to close a polygon by pressing the right mouse button.

Repeating, Canceling and UNDOing Commands

Select the following item from the menu:

1:**Again**

This will repeat the last command. The "ADD POLY" command is shown on the command line and the program is waiting for you to digitize another polygon. Instead of adding another polygon, press both the left and right mouse buttons simultaneously to cancel the command.

Another way to repeat commands is to use the arrow keys to select any command that has been executed recently in the current session. Press the < ↑ > key repeatedly until the “ADD BOX” command is shown on the command line. Now press <Enter>.

You are now creating another box. Digitize two points with the left mouse button to define the box. Note that it is created on the new current default layer rather than the layer used when you defined the first box.

Now remove the box you just added to the drawing by clicking:

1:**UNDO**

Executing **UNDO** will usually undo the last non-view command. (The 1:(VIEW)**last** menu item can be used to undo view commands. The two classes of commands are handled separately so that you can execute a command, modify the view window to examine the results, and then decide to undo the command.) UNDOing a command twice redoes the command. Click 1:**UNDO** again now. The box is recreated in the drawing.

Adding a Wire

To add a wire to layer POLY, click:

1:**UseLay → POLY → WIRE**

The menu will disappear and a red octagon with the cursor in the center will appear on the window. This is the wire cursor. Move the cursor where you want the first vertex of the wire to be placed and press the left mouse button.

Now move the cursor around in a circle without pressing any buttons and note that the wire snaps to angles that are multiples of 45°. There are snap grid and snap angle parameters that control how you can digitize points with the cursor. (These parameters are set in the startup command file described on page 48.)

Move the cursor to position the next vertex of the wire and press the left mouse button again. Now move the cursor about 1 inch past the edge of the view window. The view window will pan by one half the view window dimension.

(This feature is called autopan. Autopan is active whenever it is enabled and you are digitizing points for any command other than a VIEW command.)

Continue defining wire vertices. You can correct digitizing errors by doubling back or extending segments. After pressing the left mouse button to place the final vertex, press the **right** mouse button to tell ICED™ you have finished adding the wire.

Entering Commands on the Command Line

While the menu is shown, press the < ↑ > key once. If the "**Use Layer POLY; add wire**" commands do not appear on the command line, keep scrolling the command history with the arrow keys until they do.

The <Home>, <End>, < ← >,and < → > keys move the text insertion cursor on the command line. Press <End> now and type at the keyboard, " **width = 4**". The entire command should now look like the following:

Use Layer POLY; add wire width = 4

Press <Enter> now. You are now adding a wire with a different width. Press both mouse buttons to cancel the command.

You can type a command at the prompt any time a menu is displayed. Any input from the keyboard is automatically typed on the command line. You do not have to move the cursor to any particular location.

The SELECT IN Command

Many ICED™ commands operate on selected components and ICED™ offers a variety of ways to select them. The **SELECT IN** command selects any components whose outlines cross or are completely inside of a specified box. Select this command now by clicking:

1:(SELECT)**in**

Draw a box with the cursor that intersects one of the components that you have already created. To do this, move the cursor to a position just outside the shape and press the left mouse button. Then move the cursor so that it is inside the shape and press the left mouse button again.

When an item is selected, select marks (small white squares) are drawn on the boundaries of the component. Notice that the number of selected components is shown in the command line prompt near the lower left corner of the window after the text "Sel=".

More on Menus

When one or more components are selected, the first top-level menu is changed. Notice that a variety of UNSELECT commands, indicated by the heading UNSEL, have replaced the lower case items under the ADD heading. All of the ADD options that are now missing from the top-level menu are still available by clicking 1:**ADD**.

Copying a Component

Now that you have selected a component (it is okay if you have selected more than one), you can copy it (or them) by using the COPY command. Optionally, you can mirror components as you copy them. To perform a copy with mirroring, click:

1:**COPY → MIR Y**

The COPY MIRROR Y command will mirror the component(s) around a horizontal line drawn at a particular Y coordinate. Use the mouse to position the horizontal line and press the left mouse button to perform the copy operation.

Unselecting All Components

Now select the menu item:

1:(UNSEL)**all**

All components are now unselected.

Embedded Select Commands

The sequence of actions consisting of selecting a group of components, operating on them, and then unselecting them is very common in ICED™. Any number of components can be selected at one time. You can build a large set of selected components by combining a series of SELECT and UNSELECT commands. Furthermore, you can perform several commands on the same set of components before unselecting them. However, in simple cases, the select-operate-unselect sequence is unnecessarily cumbersome.

If no components are selected and you issue a command that operates on selected components, ICED™ will give you an opportunity to select something before executing the command. This is called an embedded select command. Embedded select commands combine the SELECT and UNSELECT commands with the command itself.

To see how this works, make sure that no components are selected by noting the "SEL=0;" remark in the command line prompt. (If a number other than 0 is reported, click 1:(UNSEL)**all**.) Click:

1:**MOVE → X&Y**

A cursor with a box will appear in the view window. Use the mouse to position the cursor box so that it crosses one side of the wire you added earlier. Press the left mouse button. The wire should now be selected and the cursor will change back to the normal cursor.

You must tell ICED™ how far to move the component by digitizing two reference points. ICED™ will move the component by the distance between them.

You already digitized the first point when you selected the component. Note that this point is marked with a white X. (If the first point is in an inconvenient location you can cancel it by pressing the right mouse button.) Move the mouse and press the left mouse button to digitize the second reference point. ICED™ will move and then unselect the wire.

Because this tutorial deals with simple geometries, you will usually use embedded select commands. It is important to remember that you can always use the select-operate-unselect sequence in more complicated situations.

Note: The exact nature of the embedded select command depends on the command you are using. In this case you used an embedded SELECT NEAR command and the box cursor is called a “near-box”.

Selecting and Stretching Parts of a Component

In addition to manipulating entire components, ICED™ can manipulate portions of a component. This is done by selecting only the sides or segments of the component you want to transform. Click:

1:(SELECT)**side**

Draw a box that intersects only one segment of the wire you have already added. You have now partially selected the wire. Move the selected segment to a new position by clicking:

1:**MOVE → X&Y**

The selected segment and the segments attached to it will be stretched or shrunk to place the segment in its new position. Move the cursor and select a point on the selected wire segment and press the left mouse button. Now move the cursor to the new position for the segment and press the left mouse button again.

To unselect the wire, click:

1:(UNSEL)**all**

You have just finished a select-operate-unselect sequence. You could have done the same thing with an embedded select command. Do this now to a different wire segment by clicking:

1:(MOVE)**side → X&Y**

Journal Files and Recovery from System Failure

Every time ICED™ completes a command, the command is recorded in the file *cell_name*.JOU in the working directory. (In this case, "Q:\ICWIN[3]\TUTOR\-MYCELL.JOU".) When you use the normal termination commands (EXIT, QUIT, or LEAVE) to terminate ICED™, this file is renamed *cell_name*.LOG.

Whenever ICED™ is launched to edit a cell, it looks to see if the file *cell_name*.JOU exists. If it does, ICED™ knows that your last session was interrupted and asks you if you want to recover. If you indicate that you want to recover your work, ICED™ will re-execute the commands in the journal file. This process is very fast, since ICED™ will not regenerate the display for each command.

[3] Remember that Q:\ICWIN represents the drive and path where you have installed ICED™.

Click the button with the 'X' in the far upper right hand corner of the window. A message box is displayed to see if you really want to close the editor without saving the cells files. Click the button labeled "JOURNAL" to proceed without saving the work you have done. This is similar to the situation you face in a power failure or when you turn off your computer without closing the editor.

Now edit cell MYCELL again. In the console window opened by the ICED icon, you can simply press the < ↑ > key to recall the last command on the command line. When “ICWIN MYCELL” is on the command line, just press <Enter>. ICED™ will display a message similar to:

> **Journal file for Q:\ICWIN\TUTOR\MYCELL exists**
> **Do you want to recover?**

Click the "YES" button. Note that the prompt at the bottom of the window tells you to press <Enter> to continue. Do so now. The journal file is executed, recovering all work you did up to the point where your session was interrupted.

(Far more complex recovery scenarios are possible with this type of recovery mechanism. An entire tutorial covers this subject beginning on page 301.)

Closing the Editor Normally

If you are using the free demo version of ICED™, the journal method described in the preceding lesson is the only method available to you to save your work. This version will not save cell files. If you close the edit session with the 'X' button in the far upper right hand corner of the window (or use the JOURNAL command), you will be able to recover all of your work the next time you open the same cell.

The purchased version will save cell files. To close the editor and save all work done during the session, use the LEAVE command. To test this command, click:

> 1:**FILE → LEAVE**

Either method will leave the console window created by the ICED icon open. If you need to close this window now, type “EXIT” at the console command prompt or click the Close button (with the X icon) in the top-right corner of the console window.

You have completed about one third of the tutorial. This is a good point to take a break.

To open the editor again, follow the steps on page 17. If you left the DOS console window open, you can simply press the < ↑ > key to recall the "ICWIN MYCELL" command and then press <Enter>.

Using the Ruler

During layout, there will be times when you need to measure the distance between two objects. This is done with the RULER command. Click:

2:**RULER**

(Remember that you must press the right mouse button (or space bar) to change to the second top-level menu. Select the **RULER** item near the middle of the second menu.)

A full window cross hair will appear. Move the intersection of the cross hair to the place you want to use as your starting point and press the left mouse button. Now, move the cross hair around using your mouse. Notice that at the bottom of the window, the current location of the cross hair, the position of the starting point, the X and Y displacements, and the diagonal distance between the two points is displayed. Press the left button again to end the use of RULER.

Nested View Commands

When editing large designs the limited resolution of the screen becomes a problem. Nested view commands provide a method of easily changing the view window from a large-scale view to a view that displays fine details during the execution of commands. You sometimes need to view a large section of the design, select a command from the menu, zoom in on a small section of the design, select a point, return to the original view, zoom in on another section of the design, select another point, and so on.

To see how this works, we will add a new box at a distance from the current geometry, then measure this distance using the RULER command and the nested view menu. First, click:

1:(VIEW)**out % → 5.0**

Add a box at the lower left corner of the view window with the menu option:

    1:(ADD)**box**

To perform the measurement, click:

    2:**RULER**

The RULER cross hair will appear, however at this scale it might be difficult to accurately measure the distance from the box to the other components.

To zoom in on the lower left corner of the view window, press the **<Esc>** key. (The center button on a three-button mouse performs the same function.) A VIEW menu with a red cursor will appear. Select **BOX**. Draw a small box that surrounds the new box in the lower left corner. Once you have digitized the view box, the window zooms in and the cursor returns to the RULER cursor allowing you to continue the interrupted RULER command. Now you can accurately place the cross hair on one corner of the box. Do so and press the left mouse button.

To return to the view at the start of the command, press the <Esc> key (or center mouse button) again. This time select **HOME**. This returns the view window to what it was when you began the current edit command.

Now press the <**Esc**> key (or center mouse button) again and use **BOX** again to zoom in on the components in the center of the window. Place the cross hair on the corner of one of the components and press the left mouse button. This completes the RULER command.

To return again to the view at the start of the RULER command, select:

    1:(VIEW)**last**

Nested view commands are useful in a variety of situations. The <Esc> key (and center mouse button) are active whenever you are selecting points for any command other than a VIEW command.

If you click the left and right mouse buttons simultaneously during a nested view command, you will only cancel the nested view command.

Deleting Selected Components

The **DELETE** command removes fully selected components from the layout. (It does not affect partially selected components.) To clear the layout, click the following menu items:

1:**SELECT → ALL**
1:**DELETE**

Now select **UNDO** from the main menu. The components will reappear. Select **UNDO** a second time to redo the DELETE command.

The TEMPLATE Command

In the exercise on page 24, you added a wire using the default width for the layer POLY. This default width parameter was set by the startup command file. You can use the TEMPLATE command to display a listing of the ICED™ parameters, including the default width for each layer. Execute this command now by clicking:

2:(TEMPLA)**screen**

A listing of ICED™ parameters will replace the view of the layout. If you do not see a group of lines all beginning with the keyword "LAYER", as shown in Figure 4, use the mouse on the scroll bar on the right edge of the window (or the < ↑ > and < ↓ > keys) to scroll through the listing until they appear.

```
LAYER 0   PEN=0
LAYER 1   NAME=NWEL    WIDTH=3.000    DIM_BLUE       PAT=1   PEN=16 …
LAYER 2   NAME=NDIF    WIDTH=3.000    GREEN   PAT=1   PEN=* …
LAYER 3   NAME=PDIF    WIDTH=3.000    YELLOW  PAT=1   PEN=* …
LAYER 4   NAME=PSEL    WIDTH=3.000    YELLOW  PAT=0   PEN=* …
LAYER 5   NAME=POLY    WIDTH=2.000    RED     PAT=1   PEN=* …
LAYER 6   NAME=M1      WIDTH=3.000    CYAN    PAT=2   PEN=* …
```

Figure 4: Portion of TEMPLATE report

Part way down this list, you will see an entry with the parameter "NAME=POLY". This line displays the parameters associated with the POLY layer. Note the default width indicated by the "WIDTH=" keyword.

When you are done viewing the report, press <Enter> (or both mouse buttons simultaneously) to return the view to the layout mode.

Adding a Wire with a Non-Default Width

You can override the default width of a new wire using menu options. First, adjust the view scale so that the red grid appears. To do this, click:

1:(VIEW)**in %** $\rightarrow$ **1.5**

This will zoom in 1.5 times the current scale. Keep clicking 1:**Again** until the red grid appears.

Please add the next two wires carefully. You will need them for the next several exercises. When you are done with this exercise your display should contain wires similar to the two wires shown in Figure 5.

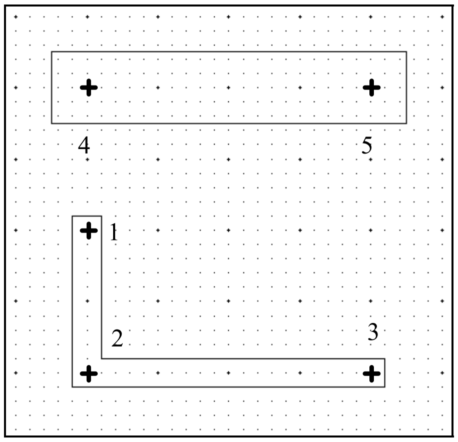


Figure 5: Adding wires.[4]

The ADD command we are about to use will add a wire on the default layer. The default layer name is always reported in the command prompt at the lower left corner of the window. Note what the default layer is right now. If it is not POLY, change it by typing the following command :

USE LAYER POLY <Enter>

Now use the ADD WIRE command to add a wire to that layer using the default width associated with it. Click:

1:(ADD)**wire**

The wire cursor appears, ready for you to digitize the coordinates in Figure 5. Move the cursor to point 1 and press the left mouse button. Then do the same for points 2 and 3. Finally, press the right mouse button to indicate that you are done adding coordinates.

Next, we will override the default layer and width. Click:

1:**UseLay** $\rightarrow$ **M1** $\rightarrow$ (WIRE)**w%** $\rightarrow$ **5.000**

[4] These graphics are drawn with FILL=OFF. Your display may fill in the outlines with color. Fill is covered later.

This will add a wire on the M1 layer with a width of 5.000. When the wire cursor appears, digitize points 4 and 5 (as shown in Figure 5) then press the right mouse button.

Note that the default layer, as reported on the command line, has changed to M1.

Displaying Component Information with SHOW

The SHOW command can be used to display data about selected components. Start by selecting the two wires you just added to the layout. To do this, click:

1:(SELECT)**in**

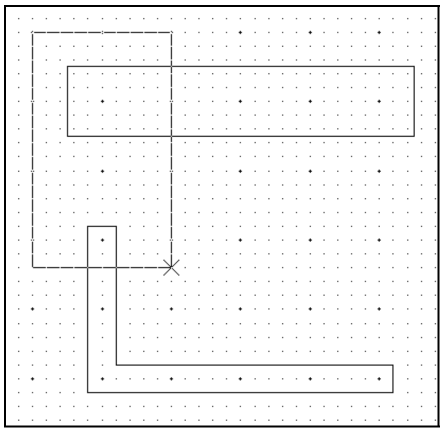


Figure 6: SELECT box used to select 2 wires.

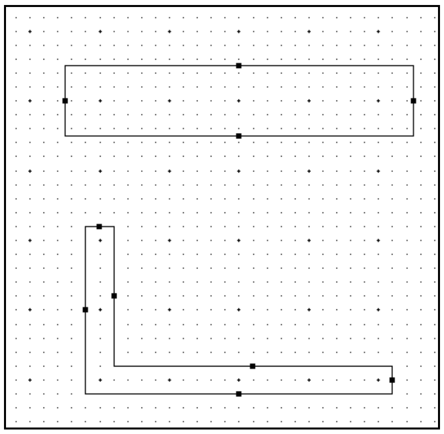


Figure 7: The 2 wires are selected.

Now, digitize a select box intersecting both wires as shown in Figure 6. White select marks will appear on the wires as shown in Figure 7. To display information on the selected components, click:

2:(SHOW)**screen**

The text for the ADD commands of the selected components will appear. (Microwave engineers should note that total wire length is reported.) Note that the LAYER and WIDTH of the two wires are different.

Press <Enter> to return the view window to layout mode.

Unselect the wires by clicking:

1:(UNSEL)**all**

Quickly Reporting the Width and Length of a Wire

The combination of an embedded SELECT command with the SHOW command allows you to display the length and area of a single wire very efficiently. Now that nothing is selected, execute the SHOW command again by clicking:

2:(SHOW)**screen**

The near-box cursor will appear in the view window. Move the cursor so that it overlaps the edge of the **'L' shaped wire**. Now press the left mouse button to select the wire and generate a display similar to Figure 8. **Note the width of the wire** as displayed on your window. (You will need to know this value in the next exercise.) Press <Enter> to return to the layout mode. Note that the wire is unselected at the end of the command.

```
! Selected components in MYCELL:
ADD WIRE LAYER=POLY ID=9 TYPE=2 WIDTH=2.000 AT (69.5, 44.0) &
        (69.5, 14.0) (125.0, 14.0)
! layer:   perimeter   area      wire length
!- POLY   191.000     452.50    90.500
```

Figure 8: Report from the SHOW command including wire area and length.

Changing the Width of a Wire with @ED

In a later exercise, we will be merging the two wires. To merge wires, they must be on the same layer and have the same width. In this exercise, we will change the width of the horizontal wire to be the same as the 'L' shaped wire.

We will do this with the Q:\ICWIN[5]\AUXIL\ED.CMD command file supplied with ICED™. This command file can be used to modify any of the parameters of a given component. A command file contains a list of ICED™ commands. These command files are executed with the *@file_name* command.

Type the following to execute this command file:

@ED <Enter>

The near cursor appears in the view window. Position the mouse on the edge of the **top wire**, then click the left mouse button to select it.

The Windows Notepad editor comes up in a new window displaying the text of the ADD command for the component. (This was generated by a SHOW command in the command file.) Now edit the WIDTH= parameter to change it to the same value as the width of the 'L' shaped wire reported in the last exercise.

To exit the text editor, type <Alt><F> and then <X>. Type <Enter> at the "Save it now?" prompt. This saves the file. The ED.CMD command file will now delete the original component and add a wire component using the ADD command you just edited. Note that the width of the two wires is now the same.

Using @UNED

After you execute a command file, the UNDO command will undo only the last command executed by the command file. To undo the effect of the entire ED.CMD command file you need to use a different method.

Type the following to execute the UNED.CMD command file:

@UNED <Enter>

[5] Remember that Q:\ICWIN represents the drive and path where you have installed ICED™.

You can see that the wire is now back to what it was before ED.CMD was executed. To redo the changes to the wire, execute UNED.CMD again by clicking:

1:**Again**

You can also execute these command files (or any command file stored in a directory known to ICED™) from the menus. You could run UNED.CMD by making the following menu choices to select the command file from lists of command files available in the directories known to the editor:

3:**@%.cmd → NextPATH →** (don't click now) **UNED**

Rather then re-execute the UNED.CMD command file, simply return to the main menu by pressing both mouse buttons.

Changing a Layer with the SWAP Command

In this exercise we will change the layer of the top wire, so that both wires will be on the same layer. We could have done this at the same time we changed the width, however, this method demonstrates a easier way to edit a component (or a selection of components) when all you want to do is change the layer.

The easiest way to change the layer of a component is to use the SWAP command. Click the following menu options:

2:(SWAP)**layers → POLY → M1**

The menu disappears and the near cursor appears in the view window for you to select the horizontal wire at the top of your window. Once you select the wire, its layer is changed from M1 to POLY. You can test this with the SHOW command if you want to.

(The SWAP command is more powerful than a simple "change layer" operation. When components on both indicated layers are selected, they all will have their layers swapped. You can also swap cells with options of the SWAP command.)

Now that both wires have the same width and are on the same layer, we can merge them.

Merging Two Wires

The MERGE WIRES command connects two selected wires to form a single wire. **Both wires must be on the same layer and have the same width.**

In order to merge two wires, you must **partially select** them. Each wire must have one selected end and one unselected end. The selection of the other sides of the wires is not important.

Your display should look something like Figure 5 on page 34. Make sure that no components are selected, then perform the merge with the following menu options:

1:(UNSEL)**all**
2:(MERGE)**wire**

You are now executing an embedded SELECT END IN command. Use the mouse to digitize the same box shown in Figure 6 on page 35. However, since this is a SELECT END IN command rather than a SELECT IN command, only the ends of the wires in the box are selected.

The MERGE command will extend, or contract, the selected ends of both wires and merge them to form a single wire as shown in Figure 9.

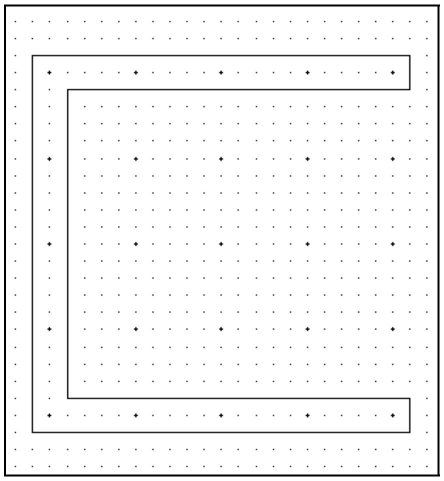


Figure 9: Merged wire.

It is not always possible to select both wire ends at the same time without selecting an extra component. If this is the case, you must select them prior to executing the MERGE command. To see how this is done, select 1:**UNDO**. Then choose 1:(SELECT)**end**. Use the mouse to digitize a box that surrounds the top of the L-shaped wire. This will select the top end of the wire. Now select 1:**Again**. Use the mouse to digitize a box that surrounds the left side of the horizontal wire. Now select 2:(MERGE)**wire** again. Watch what happens.

The FILL Command

ICED™ has two display modes, FILL ON and FILL OFF. You can toggle between them by selecting 3:(FILL)**tog** or you can just type the following abbreviation for the FILL command at the command prompt:

F <Enter>

When you type commands at the prompt, you can always abbreviate keywords as long as the abbreviation is unambiguous. No other command begins with the letter 'F' so typing simply <F><Enter> is sufficient to execute the FILL command.

Click 1:**Again (**if necessary) to toggle the fill mode to off.

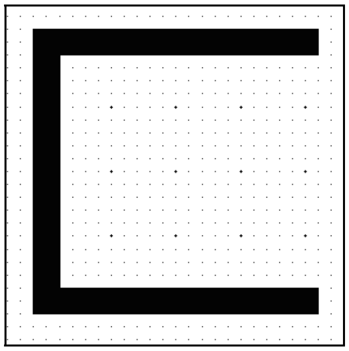


Figure 10: Wire drawn with FILL on.

Merging a Wire and a Box

The MERGE WIRES command can also be used to merge a wire and a box, as long as they are on the same layer and the box has the same width as the wire. The wire must have one end selected and the box must have either one or three sides selected.

To demonstrate this type of merge, we first need to add a box with the appropriate width. Click:

1:**UseLay→ POLY → BOX**

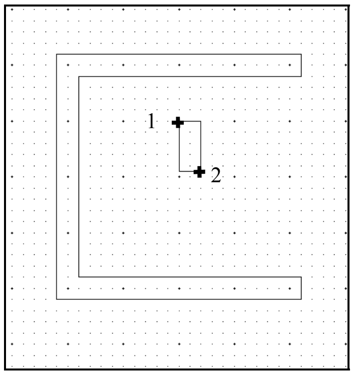


Figure 11: Adding box.

Digitize the coordinates carefully with the cursor. See Figure 11. Use the report on coordinates at the bottom of the window as you are digitizing point 2 to insure that the width of the box is exactly the same as the width you noted in the earlier exercise.

Now we will merge this box with the wire. Click:

1:(SELECT)**side**

Now use the cursor to define a selection box that intersects one end of the box and one end of the wire. See Figure 12. It is important that you partially select only one end of each shape. To perform the merge, click:

2:(MERGE)**wire**

The result of this operation is shown in Figure 13. Note that the end of the top wire segment has been trimmed back to perform the merge.

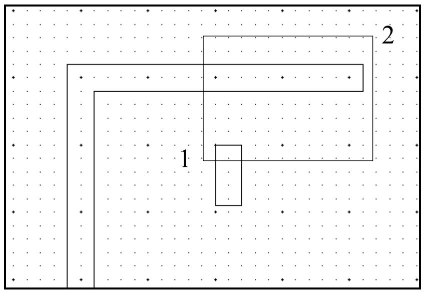


Figure 12: Selecting end of box and wire.

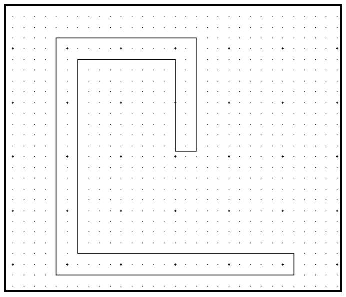


Figure 13: Result of merge.

Cutting a Wire

The CUT command is used to divide wires, polygons, or lines into two parts. To see how the CUT command works, click:

2:(CUT)**hori-Y**

You are now executing an embedded SELECT SIDE IN command. Select only the single vertical segment of the wire as shown in Figure 14.

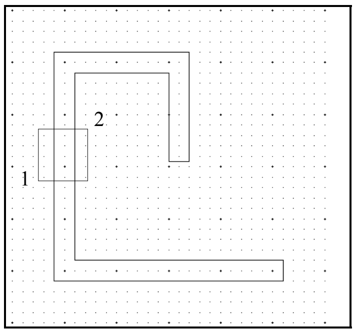


Figure 14: Selecting a segment of a wire.

A horizontal cut-line will appear. (A horizontal line is defined by the equation Y=*constant*. This leads to the menu label, hori-Y.) The wire will be cut where the line intersects the selected segment of the wire. Position the cut line with the cursor so it intersects both vertical segments. Now click the left mouse button.

Even when the cut line intersects more than one wire segment, only the selected segment is cut. If the cut line intersects more than one selected segment, the cut will fail.

Merging Polygons

The MERGE POLYGONS command is used to merge two polygons to form a single polygon. The two polygons **must touch** along at least one side. Either or both polygons may actually be boxes. If the polygons touch each other from the outside, the merged polygon is the sum of the two polygons. If they share a common area as well as a common side, the merged polygon is the difference of the two polygons. Thus, the MERGE command can be used to add or remove pieces of an existing polygon.

Hold the <Ctrl> key and press the < ↑ > key twice to move the view window to a clean section of the drawing. Next, add two boxes as shown in Figure 15 with:

1:(ADD)**box**
1:**Again**

To perform the merge operation, click:

2:(MERGE)**poly**

With the near-box cursor, select the common side as shown in Figure 15. The merged polygon should look similar to the one shown in Figure 16.

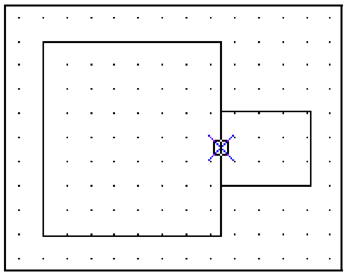


Figure 15: Selecting the common side of 2 boxes.

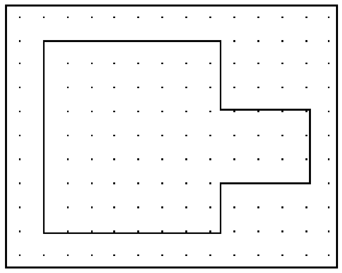


Figure 16: Merged polygon.

Cutting a Polygon

The CUT command can also be used to cut a polygon. To perform this operation, click:

2:(CUT)**vert-X**

Digitize a select box that cuts the top and bottom sides of the polygon as shown in Figure 17. A vertical cut-line will appear and 2 sides of the polygon will be indicated with select marks as shown in Figure 18. The cut-line must intersect exactly two selected sides of a polygon. If it intersects more or fewer sides, the cut will fail. Press the left mouse button to digitize the line position and cut the polygon.

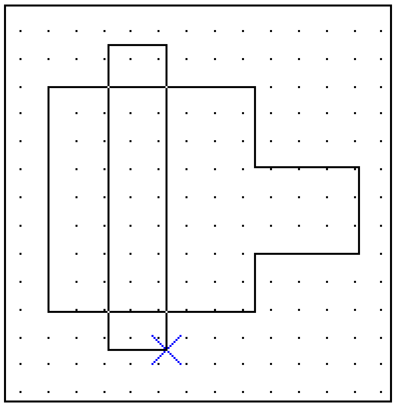


Figure 17: Selecting sides of a polygon.

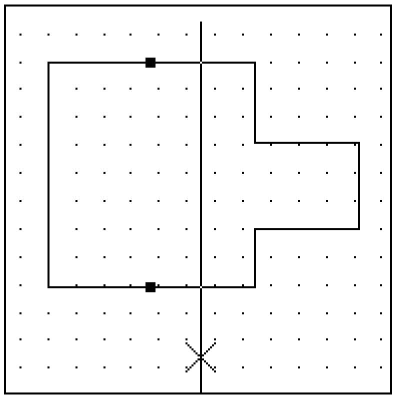


Figure 18: Using cut line to cut the indicated sides of a polygon.

Adding Cells

Hold the <Ctrl> key and press the < ↑ > key twice to reach an unused area of the drawing. Now click:

1:(ADD)**cell**

A cell menu will appear. This menu should list the sample cells in the directory, Q:\ICWIN[6]\SAMPLES. The cell directory name is reported on the history line. (If the SAMPLES directory is not indicated, select the **NextPATH** menu item until it is.)

Select the cell named VIA by moving the cursor to the line **VIA** and pressing the left button on your mouse. After selecting the cell name, you will see the outline of a box displayed on the window. This is an outline of cell VIA. Move the mouse to position the cell then press the left button to place the cell and end the command.

Now let us add another cell. Select the ADD CELL command again:

1:(ADD)**cell**

Note that the first cell list displayed has changed. If you already have cells added to the current drawing, ICED™ will display those cells in the first list. To see cells in the next cell library, select **NextPATH.**

If the Q:\ICWIN\SAMPLES directory is not currently listed, select **NextPATH** from the menu until it is. Now select **NAND**. A cell outline will appear. Press the left mouse button to add the cell to your drawing.

Now let us add another copy of the NAND cell, but we will mirror this copy as we add it. Select the ADD CELL command with the mirror option by clicking:

1:**ADD** → (Cell %)**mY** → **NAND**

To add more copies of the same cell, you can use the 1:**Again** menu option. Add a few more copies of this cell. Change the view menu during the command if necessary with the nested view menu by pressing <Esc> or the center button on a three-button mouse.

Ungrouping a Cell

A cell is just a group of components. A cell can be ungrouped so that you can select and modify individual components without modifying other copies of the cell. To see how this works, click:

2:**EDIT** → **UnGrp**

[6] Remember that Q:\ICWIN represents the drive and path where you have installed ICED™.

You are now executing an embedded SELECT CELL * AT command. Move the cursor inside the white dotted line that surrounds a copy of the NAND cell. (When using the SELECT CELL * AT command, placing the cursor on the edge of a cell will not select it.) Press the left mouse button to ungroup the cell. The dotted white cell outline will disappear. This indicates that the individual components can now be operated on like any other component in the drawing. The other copies of NAND are not affected.

Selecting New Components

The SELECT NEW command generally selects components that were changed by the last non-VIEW command. Select:

1:(SELECT)**new**

Notice that the components of the ungrouped cell are selected.

Unselecting Components by Layer

You can select/unselect components by layer. We will unselect components in our ungrouped cell except for the M1 shapes. Click:

1:(UNSEL)**layer → M1**

Note that the selected layer name turns yellow. You could continue to select more layers at this point. Now click:

Invert

Now the M1 layer name returns to a green color, and all of the other layer names turn yellow indicating that all layers except for M1 will be unselected. To indicate that you have finished selecting layers and continue the command, press the **right** mouse button (or click Return). Select **ALL** from the next menu.

Constraining Moves to one Direction

Move the M1 components in the ungrouped cell down slightly with the option:

1:**MOVE → Y**

This allows the selected components to move only vertically. Now you must digitize two points to define a displacement vector. Place the cursor anywhere on the screen and press the left button once. Now move the cursor down slightly. Note that the exact displacement is reported at the bottom of the ICED™ window in the following format:

(*current location*) -(*first coordinate*) = (*disp_x*, *disp_y*)

Now press the left mouse button to digitize the second coordinate of the move. Note that only the selected components have moved. Note also that the components moved straight up or down. The displacement in the x-direction was ignored.

Using GROUP to Create a Cell

The GROUP command takes fully selected components and creates a new cell from them. The selected components are removed from the drawing, and the new cell is added in their place.

First, we must select all components in our modified NAND circuit. We will use the SELECT IN command for this. Click:

1:(SELECT)**in**

Use the cursor to draw a select box around the modified NAND circuit. This will select the unselected components again. Components that were already selected are not affected.

To remove these components from the layout and add a newly created cell in their place, select:

2:**EDIT →** (GROUP)**at ***

Note the prompt near the bottom of the window and type the cell name as "NANDTEST". Then press <Enter>. You have now created a cell with the name NANDTEST. The origin of this new cell is at the lower left-hand corner of the bounding box indicated by the dotted white line around the cell.

The new cell file (NANDTEST.CEL) will be stored in the working directory, Q:\ICWIN\TUTOR, when you exit the editor (unless you are using the demo version of ICED™).

Aligning Cells

Since our modified NANDTEST cell has had its buss wires shifted, it will need to be aligned with the busses in other cells for the busses to line up. We can align components easily with the MOVE Y (or MOVE X) commands.

First select the entire cell by clicking:

1:(SELECT)**new**

Now begin the MOVE operation with:

1:**MOVE → Y**

Use the cursor and the left mouse button to digitize a reference point on the lower edge of the lowest M1 wire in the NANDTEST cell. Now digitize the second reference point on a lower edge of a similar wire of one of your copies of the NAND cell. The cell moves vertically to align the bus wires, displacement in the x-direction is ignored.

Exiting ICED™

If you are using the demo version, terminate the editor with the QUIT command:

1:**FILE → QUIT**

If you have the full version, exit ICED™ and save your work by selecting:

1:**FILE → LEAVE**

All modified cell files are saved at this time. The LEAVE command saves only cell files whose components have been changed. The EXIT command saves all open cell files whether or not they have changed.

The LEAVE command is generally preferable to the EXIT command, since it leaves the date stamp of unmodified cells unchanged. However, if you edit a cell and change only environment settings (e.g. display or layer parameters), but change no geometry, those new environment settings are lost when you close the editor with the LEAVE command.

The Startup Command File

Back on page 18, you learned that when the layout editor opens to create a new cell, it executes a startup command file. This startup command file is a command file like ED.CMD. It contains a list of layout editor commands.

You can copy the sample startup command file to a new location and edit the commands in it to customize options like colors, layer names, grids, etc. This operation is beyond the scope of this tutorial, but we explore this subject in the next tutorial.

The specification for the startup command file is defined with the START option in the command line stored in the ICWIN.BAT file. The specification in ICWIN.BAT looks like:

START=Q:\ICWIN[7]\TECH\SAMPLES\NEW.CMD

The contents of this command file are shown below.

```
VIEW OFF
$ MENU "M1";  KEEP_LIBRARY_CELLS=ASK;  DISPLAY CELL_DEPTH=100;
PATTERN "SAMPLE";  FILL MIXED ON
AUTOPAN ON  PIXELS=100  SECONDS=0.5;   ARROW MODE=EDIT
DISPLAY CELL_LABELS=ON OUTLINE_DEPTH=1 EDIT_STACK=OFF CURSOR 1 SNAP=ON
SPACER OFF  SPACE=0.0  TRACK_LAYERS=OFF  STYLE=1
VIEW LIMIT ON   SCALE=0.500 DEPTH=1 DOTS=0 UNITS=0.0 SHOW_LAYERS 1:100
ARRAY  DRAW_MODE=SIDES  CELL_LIMIT=1024
TEXT LOWER_CASE=DISABLED MULTI_LINE_TEXT=DISABLED ORIENT=2  DISPLAY_ORIGINS=OFF
USE TEXT_JUSTIFICATION=LB  WIRE_TYPE=2  ARC_TYPE=2  N_SIDES=16
RESOLUTION  STEP=0.500  MODE=SOFT
SNAP  ANGLE=45  STEP=(0.500,0.500)  OFFSET=(0.000,0.000)
NEAR  UNITS=0.05  DOTS=4
```

(Continued on next page.)

[7] Remember that Q:\ICWIN represents the drive and path where you have installed ICED™.

```
COLOR  0 NAME=BLACK       PALETTE=( 0, 0, 0)
COLOR  1 NAME=WHITE       PALETTE=(63,63,63) LEVEL=16
COLOR  2 NAME=YELLOW      PALETTE=(63,63, 0) LEVEL= 6
COLOR  3 NAME=GREEN       PALETTE=(21,63, 0) LEVEL= 6
COLOR  4 NAME=RED         PALETTE=(63, 0,21) LEVEL= 8
COLOR  5 NAME=CYAN        PALETTE=( 0,42,42) LEVEL= 9
COLOR  6 NAME=BLUE        PALETTE=( 0, 0,63) LEVEL=10
COLOR  7 NAME=MAGENTA     PALETTE=(63, 0,63) LEVEL= 8
COLOR  8 NAME=GRAY        PALETTE=(42,42,42) LEVEL=14
COLOR  9 NAME=BROWN       PALETTE=(32,16, 0) LEVEL= 8
COLOR 10 NAME=ORANGE      PALETTE=(63,31, 0) LEVEL= 8
COLOR 11 NAME=PURPLE      PALETTE=(21, 0,14) LEVEL= 3
COLOR 12 NAME=DIM_RED     PALETTE=(22, 0, 0) LEVEL= 3
COLOR 13 NAME=DIM_BLUE    PALETTE=( 0, 0,22) LEVEL= 3
COLOR 14 NAME=DIM_GREEN   PALETTE=( 0,22, 0) LEVEL= 3
COLOR 15 NAME=HI          PALETTE=(63,63,63) LEVEL=15

COLOR BLACK  ONE_DOT=WHITE    FOUR_DOTS=(WHITE,  WHITE,  WHITE,  WHITE)
COLOR WHITE  ONE_DOT=BLACK    FOUR_DOTS=(BLACK,  BLACK,  BLACK,  BLACK)
COLOR YELLOW ONE_DOT=YELLOW   FOUR_DOTS=(YELLOW, YELLOW, YELLOW, YELLOW)
COLOR GREEN  ONE_DOT=GREEN    FOUR_DOTS=(GREEN,  GREEN,  GREEN,  GREEN)
COLOR RED    ONE_DOT=RED      FOUR_DOTS=(RED,    RED,    RED,    RED)
COLOR CYAN   ONE_DOT=CYAN     FOUR_DOTS=(CYAN,   CYAN,   CYAN,   CYAN)
COLOR BLUE   ONE_DOT=BLUE     FOUR_DOTS=(BLUE,   BLUE,   BLUE,   BLUE)
COLOR MAGENTA ONE_DOT=MAGENTA FOUR_DOTS=(MAGENTA,MAGENTA,MAGENTA,MAGENTA)
COLOR GRAY   ONE_DOT=BLACK    FOUR_DOTS=(BLACK,  WHITE,  WHITE,  WHITE)
COLOR BROWN  ONE_DOT=RED      FOUR_DOTS=(GREEN,  RED,    RED,   YELLOW)
COLOR ORANGE ONE_DOT=RED      FOUR_DOTS=(RED,    YELLOW, YELLOW, RED)
COLOR PURPLE ONE_DOT=MAGENTA  FOUR_DOTS=(BLUE,   MAGENTA,MAGENTA,BLUE)
COLOR DIM_REDONE_DOT=RED      FOUR_DOTS=(RED,    WHITE,  WHITE,  WHITE)
COLOR DIM_BLUE ONE_DOT=BLUE   FOUR_DOTS=(BLUE,   WHITE,  WHITE,  WHITE)
COLOR DIM_GREEN ONE_DOT=GREEN FOUR_DOTS=(GREEN,  WHITE,  WHITE,  WHITE)
COLOR HI     ONE_DOT=BLACK    FOUR_DOTS=(BLACK,  BLACK,  BLACK,  BLACK)

GRID 1  ON   COLOR=RED     DOTS     STEP=1.0
GRID 2  ON   COLOR=CYAN    CROSSES  STEP=5
GRID 3  OFF  COLOR=WHITE   LINES    STEP=50000

LAYER *  WIDTH=2.0  SPACE=0.0  YELLOW  PAT=0   NO_PEN
INITIALIZE LAYERS 0:255
LAYER 0 PEN=0
LAYER 1 NAME=NWEL WIDTH=3.000 SPACE=0 DIM_BLUE PAT=1 PEN=16 NO_CIF NO_STREAM
LAYER 2 NAME=NDIF WIDTH=3.000 SPACE=0 GREEN    PAT=1 PEN=* NO_CIF NO_STREAM
LAYER 3 NAME=PDIF WIDTH=3.000 SPACE=0 YELLOW   PAT=1 PEN=* NO_CIF NO_STREAM
LAYER 4 NAME=PSEL WIDTH=3.000 SPACE=0 YELLOW   PAT=0 PEN=* NO_CIF NO_STREAM
LAYER 5 NAME=POLY WIDTH=2.000 SPACE=0 RED      PAT=1 PEN=* NO_CIF NO_STREAM
LAYER 6 NAME=M1   WIDTH=3.000 SPACE=0 CYAN     PAT=2 PEN=* NO_CIF NO_STREAM
LAYER 7 NAME=M2   WIDTH=4.000 SPACE=0 BLUE     PAT=3 PEN=* NO_CIF NO_STREAM
LAYER 8 NAME=CONT WIDTH=2.000 SPACE=0 WHITE    PAT=1 PEN=* NO_CIF NO_STREAM
LAYER 9 NAME=VIA  WIDTH=3.000 SPACE=0 GRAY     PAT=1 PEN=* NO_CIF NO_STREAM
GLOBAL KEY.CF9="@UNED"
GLOBAL KEY.F1="RULER"
GLOBAL KEY.F7="DOS"
GLOBAL KEY.F8="@DEEPSHOW"
GLOBAL KEY.F9="DOS"
```

Figure 19: NEW.CMD startup command file

Conclusion

This concludes the Editor Basics tutorial. You now know enough to create cells and modify geometry. Feel free to experiment with the menus to explore the features of the layout editor.

Please read the following additional tutorials to help you master other features of the layout editor. These other tutorials are shorter and more focused than this first one. If you take a little more time to learn about advanced features, you will save many hours of work and frustration in the future.

Customizing ICED™ for Specific Projects

This tutorial will cover the methods used to customize the layout editor to define technology-specific and project-specific parameters.

Technology-specific parameters include layer names, colors, grid settings, etc. You may be familiar with the “technology files” used to define parameters like these in other layout software. In the ICED™ layout editor, you define these technology-specific parameters in a file called a **startup command file**. Typically you create a separate startup command file for each process you use. For example, you would need separate startup command files for a 0.25µ digital CMOS process and a 1.0µ analog bipolar process. Several projects that use the same technology may share a single startup command file.

Project-specific parameters include the search paths for subcells and command files and the name of the startup command file. These settings are defined in a separate file. In other layout programs it is often referred to as the “project file”. In ICED™, project specific parameters are usually defined in the batch file used to open the editor, usually referred to as a **project batch file**. The sample project batch file included with the editor distribution is named **ICWIN.BAT**. Depending on circumstances, you may need a different batch file for each project.

You must customize copies of **both** of these files to tailor the behavior of the editor **before** you use the editor to create new cells in a real design.

Changes in technology parameters midway through a project can cause problems. Changes in layer numbers, layer names, or grid resolution are particularly troublesome. Changes to project-specific search paths can also cause unfortunate results, especially if you wind up using obsolete copies of some cell files, or if two people working on the same project wind up using different cell libraries. For these reasons, it is important to understand the material in this tutorial before you begin to use the editor for real work.

Overview of Startup Command Files

We refer to the technology-specific initialization file as a startup command file since it is a file of layout editor **commands** executed automatically at the **start** of the editor session when you create a new cell.

The startup command file is where you define technology-specific parameters such as grid resolution and layer properties. Parameters that control the behavior of the editor are also set here, e.g. color definitions, display parameters, and keyboard shortcuts.

Once these types of settings are defined with the startup command file, they are stored in the cell file. We refer to these settings collectively as the **environment database** of the cell.

As you add components to the cell, their definitions are stored in the **geometric database** of the cell. It is the **layer number** of each component that is stored in this database.

The environment database stores the layer name assigned to each layer number. This environment database can be easily replaced in cells, and changes to layers names do occur in real projects. (See page 61.) However, since most people add or modify layers by the more easily remembered layer names, it is best to insure that all cells in a project have the same layer names assigned to each layer number before any cells are created. It is the startup command file that initializes this layer name - layer number correspondence in the environment database of new cells.

You should prepare the startup command file **before** creating real cells. The startup command file will be automatically executed as you start the editor when you:

- create a new cell
- open an existing cell file that does not have an environment database because it was created by the UNSTREAM (or UNCIF) import utility, or
- open a cell created by an obsolete version of ICED™ so old that the environment database structure has changed (very rare).

Note that opening an existing cell does not normally result in automatic execution of the startup command file. **Changed settings in a startup command file are not automatically reflected in existing cells.** Changing a startup command file after some cells are created may result in conflicting definitions in different cells.

There are many other important settings in a startup command file aside from the layer name – layer number correspondence. See the listing of the sample startup command file supplied with the installation (Q:\ICWIN[8]\TECH\SAMPLES-\NEW.CMD) on page 49.

[8]Throughout this manual, Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. If you have installed the software on your C drive in the directory \ICED, you should replace Q: with C: and \ICWIN with \ICED.

The usual method used to create a new startup command file is to copy the sample startup command file and edit it with any text editor. We will be editing several copies of the sample file in lessons below.

We recommend that you never alter the original startup command file supplied with the installation. Leave this file intact for future reference and for use as a template in creating your own startup command files. You may want to change the permission setting for this file to read-only.

Where to Locate Technology Specific Files

The startup command file may be only one of several technology-specific command files used in your design. It is best to locate all of these files together in one directory used only for this purpose. We suggest making a separate technology subdirectory for each technology you use. **Do not make copies of startup command files in cell libraries.** Users who make copies of the startup command file in cell libraries frequently have problems when they alter one copy but do not realize that an old copy is executed instead because it is first on the search list.

If multiple users will be creating cells using this technology, it is best to locate the technology directory on a shared network drive, so that everyone is using the same files. It is a good idea to fully qualify the startup command file name with the directory path in the editor command line in the batch file, rather than letting the editor search for it. This insures that everyone uses the same startup command file. (You'll see how to do this a little later on.)

Overview of ICWIN.BAT

The recommended way to launch the layout editor is to use a project batch file. This batch file assigns various search paths to certain **environment variables**. These environment variables are stored by the operating system and are available to all ICED™ programs and utilities executed from the console window. The layout editor uses the directory names stored in the environment variables to look for cell files and other files used by the program.

The batch file also contains the **program command line** that launches the layout editor. Several command line parameters are usually included to customize settings such as memory usage. The name of the startup command file is one of these parameters.

A sample batch file is supplied with the installation, Q:\ICWIN[9]\ICWIN.BAT. If you worked through the basic tutorial, you opened the editor by executing this batch file. Look at the listing of this file below.

```
cls
set iced_home=q:\icwin;
set iced_path=q:\icwin\samples;
set iced_path_1=
set iced_cmd_path=q:\icwin\tech\samples;
set drc_path=q:\icwin\tutor;
if _%1==_ goto done
q:\icwin\icwin %1 start=q:\icwin\tech\samples\new menu=* pause=0 maximize=yes win=500
:done
```

Figure 20: The Q:\ICWIN\ICWIN.BAT file

The first line is simply a DOS command that clears the console window. The next 5 lines use the DOS command SET to define environment variables. The purpose of each environment variable is listed in Figure 21.

The next two lines of the batch file use the %1 parameter to make the batch file multipurpose. The DOS command interpreter will replace the "%1" parameter with the cell name if you provide the cell name when you execute the batch file.

This batch file can be used to set the environment variables and then quit without opening the editor. The "if _%1==_ goto done" line handles this situation. When you want to execute the batch file to only create the project environment for other programs, you execute the batch file with no argument. In this case, the environment variables are defined and the batch file terminates. (The DRC and NLE programs, and the ICLIST and ICTREE utilities, all make use of the environment variables.) Execute the batch file prior to running these programs in the same console window if these programs will need the information stored in the environment variables.

[9] Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. Replace Q: and \ICWIN with the appropriate values.

The batch file is usually executed with a single argument that represents the name of the cell you wish to open. This cell name replaces the "%1" parameter in the layout editor command line. Several other parameters are included on the command line including the name of the startup command file. This command line can contain many other parameters. A complete description of all command line parameters is included in the Command and Utility Reference Manual.

Environment variable	**Purpose**
ICED_HOME	Base location of the directory tree containing support files
ICED_PATH, ICED_PATH_*n*	Cell library search path (The _*n* variables allow long paths to be split over several lines.)
ICED_CMD_PATH ICED_CMD-_PATH_*n*	Search path for command files (The _*n* variables allow long paths to be split over several lines.)
DRC_PATH	Search path for DRC (Design Rules Checker) rules files

Figure 21: ICED™ environment variables

You can add other DOS commands to a project-specific batch file. For example you can add a CD command to make the working directory the current directory. (We will customize one of our sample batch files in this manner on page 74.) In this case you will not need to explicitly make the working directory the current directory before executing the batch file. However, having the CD command in the batch file makes the batch file useful only to open the editor to edit cells in this working directory. You will not be able to use this batch file to open the editor to edit or browse cell files in other directories. (Once the editor is open, you can open any of the cells in one of the cell libraries with the subcell edit commands.)

The usual method to create a batch file for a new project is to copy the sample batch file supplied with the installation to a new file, then edit it appropriately. **It is strongly recommended that you keep all batch files used to launch the editor in the Q:\ICWIN directory.** Executing obsolete copies of batch files has caused headaches for ICED™ users, especially when several copies existed in various directories. For this reason, we urge you to resist the temptation to locate the batch files in cell libraries.

We will cover several examples of batch files in lessons below. All steps in copying and modifying the files will be explained. If you make an editing mistake, or if you exceed available environment space (only a problem with Windows 95[10]) then the

[10] If you exceed available environment space in Windows 95, bring up the Properties menu for the console window and use the Memory tab to increase the Initial Environment parameter.

batch file will fail. Look carefully at the message from the operating system to fix the problem.)

How the Editor Loads Cells

Recall that the ICED™ editor saves the data base of each cell in a separate cell file (*cell_name*.CEL). A directory containing cell files is called a **cell library**.

When you launch the editor, you always supply the name of a cell file on the ICED™ program command line. We refer to this cell as the **root cell**. You may edit other cells during the edit session. (We explain how to do this in the Exploring Cell Hierarchy Tutorial beginning on page 153.)

The directory containing the root cell file is referred to as the **working directory**.

You can specify the complete path to the cell file on the command line. For example, if you launch the editor by typing:

```
ICWIN C:\CHIPS\ALU\ADDER
```

The editor will try to open ADDER.CEL in the directory C:\CHIPS\ALU. In this case the working directory is C:\CHIPS\ALU, the root cell is ADDER, and the root cell file is ADDER.CEL.

It is usually more convenient to make the working directory your current directory before launching the editor with the batch file. In this method you might type:

```
CD \CHIPS\ALU
ICWIN ADDER
```

In any event, **the editor searches only the working directory for the root cell file**. If the file does not exist in this directory, a new cell is created. In this case, the startup command file is executed to create the environment database for the new cell.

If the cell file is found in the working directory, both the environment database and geometric database of this cell are loaded into the editor. Even if other cells are opened during the edit session, their environment databases are ignored. **The environment of the root cell is the only environment database loaded during the editor session.** The current environment settings in the editor are stored in each of the saved cell files, replacing any existing environment settings.

If the root cell contains subcells, the editor must search for these cell files. The editor will first search the working directory for each these cell files. If a cell file is not found there, the editor continues the search in each of the cell libraries on the **ICED_PATH** search path defined in the batch file.

The cell libraries listed in ICED_PATH are searched in the order in which they are listed in the variable. As the editor searches for a cell, only the first cell with that name found on the ICED_PATH search path is loaded, other versions in other cell libraries are ignored. If a subcell's cell file is not found, the editor issues an error message and terminates.

There is no way to change the working directory or the cell library search path once editor is open.

Where to Locate Cell Libraries

ICED™ is very flexible about where cell files are stored. You can locate cell files in any convenient directory, including directories located on networked computers. (The only locations you should avoid when storing cell files are the main Q:\ICWIN directory and other directories already used to store other ICED™ support files.)

You create a directory for cell library in the same way you create any directory. We recommend that you **avoid using blanks in cell library directory names**. Blanks in directory names can cause problems for many command files.

We strongly recommend against storing cell files in the main Q:\ICWIN directory.

A Trivial Example

Had enough of the lecture? Let us actually perform a simple project and technology customization, and you will see how easy this process really is. The files and directories you will create are shown in Figure 22.

Q:\ICWIN\	installation directory for the ICED™ software
icwin.bat	sample project batch file
ictriv.bat	new project batch file for the TRIVIAL example
TECH\	main directory for technology-dependent files
SAMPLES\	sample process-specific files supplied with ICED distribution
new.cmd	sample startup command file
TRIVIAL	new subdirectory for the TRIVIAL process-specific command files
trivnew.cmd	new startup command file for the TRIVIAL process
TUTOR\	parent directory for all tutorial cell files
TRIVWORK	working directory for TRIVIAL example
TRIVLIB	cell library for the TRIVIAL process
colors.cel	cell files (copied from Q:\ICWIN\SAMPLES)
nand.cel	
nn.cel	
pp.cel	
via.cel	

Figure 22: Files and directories used in the TRIVIAL example. Directory names are in upper case. File names are in lower case. The names of objects created in this tutorial are shown in **bold** characters.

Open a console window with the ICED icon created on your desktop by the installation. Create a new technology directory to store the modified startup command file. (We provide the DOS method to do this below, but you can use whatever method you are comfortable with to accomplish this.) Type the following in the console window:

```
CD TECH
MD TRIVIAL
CD TRIVIAL
COPY Q:\ICWIN[11]\TECH\SAMPLES\NEW.CMD TRIVNEW.CMD
```

[11] Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. Replace Q: and \ICWIN with appropriate values.

Now edit this copy of the startup command file. If you are familiar with a Windows text editor, use that program to edit the file and leave the file open in a separate window. If you do not have a different favorite editor, you can use the NOTEPAD.EXE program by typing the following in the console window:

```
NOTEPAD TRIVNEW.CMD
```

Near the bottom of the file are the layer definitions. Find the line that assigns the name M2 to layer number 7. This same line assigns the color BLUE to this layer. Change the color parameter to WHITE as indicated below

```
Old:  LAYER 7  NAME=M2  WIDTH=4.000  SPACE=0.000  BLUE …
New:  LAYER 7  NAME=M2  WIDTH=4.000  SPACE=0.000  WHITE …
```

This is the only change we will make to this file. **Save the file** but leave the file open. (Use the File drop-down menu and click the Save option.)

Create a new working directory by typing the following in the console window:

```
CD \ICWIN
CD TUTOR
MD TRIVWORK
```

Create a new cell library directory and copy some cells into this library by typing:

MD TRIVLIB

CD TRIVLIB

COPY \ICWIN[12]\SAMPLES*.CEL

The TRIVLIB directory will not be our working directory, it is a cell library defined for the layout editor to search for subcells.

Now copy and open the batch file.

```
CD \ICWIN
COPY ICWIN.BAT ICTRIV.BAT
NOTEPAD ICTRIV.BAT
```

[12]Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. Replace Q: and \ICWIN with the location you defined during the installation.

In the window opened to edit the copied batch file, find the line that defines the cell library path and change it as indicated to point to the new cell library. (You can type in upper or lower case. Either way, all statements are translated to upper case.)

Old: set iced_path=q:\icwin\samples;

New: set iced_path=q:\icwin\tutor\trivlib;

Also change the command line parameter to point to our new startup command file.

Old: q:\icwin\icwin %1 start=q:\icwin\tech\samples\new menu=* …

New: q:\icwin\icwin %1 start=q:\icwin\tech\trivial\trivnew menu=* …

Save the file and close the window.

Now make the working directory the current directory before we open the layout editor.

CD TUTOR\TRIVWORK

(If the TRIVWORK directory already existed from a previous implementation of this tutorial, cell files may already exist in this directory. If this is the case, clear the directory by typing the following.)

DEL *.*

Open the layout editor to create a new cell using the new project batch file:

ICTRIV NEWCELL

Now that the layout editor is open, let us add a cell from our cell library. Use the menu option:

1:(ADD)**cell** [13]

The menu is changed to list cells in the first cell library. Look at the history line to see which cell library is listed. It should read

```
Q:\ICWIN\TUTOR\TRIVLIB
```

Now select one of the cells we copied into our test cell library with the menu option:

NAND

[13] This menu notation was explained in the first tutorial. The number before the colon indicates the number of the top-level menu. The string in parentheses indicates the heading in the menu. The bolded string is the menu item you click with the mouse.

Click the left mouse button to place the cell. Now add a box on the M2 layer to the design. (Remember that you changed the color of this layer to white in the startup command file.)

1:**UseLay → M2 → Box**

Define two corners of the box with the mouse. Then close the editor and save the cell file with the menu option:

1:**FILE → LEAVE**

Changing a Technology Parameter in the Startup Command File

Let us suppose that you now decide that you do not like the color white for the M2 layer. Change the startup command file to change this color to brown. You should still have this file open. If not, open the file "q:\icwin\tech\trivial\trivnew.cmd" again.

```
Old:  LAYER 7  NAME=M2  WIDTH=4.000  SPACE=0.000  WHITE …
New:  LAYER 7  NAME=M2  WIDTH=4.000  SPACE=0.000  BROWN …
```

Save the file, but leave the file open. Now edit NEWCELL again by repeating the last command in the console window.

ICTRIV NEWCELL

You should see that the box on M2 is still white! Since this is an existing cell, the startup command file was not executed. The color of the M2 layer remains as it was created by your previous version of the startup command file.

If you had created a new cell, rather than opening an existing cell, then the color of the M2 layer in the new cell would have been brown. In this case you would have conflicting layer definitions in the two cells. The problem of conflicting colors is both obvious and harmless. However, if your change to the startup command file had been to change the layer number of the M2 layer, and a deeply nested library cell had a conflicting definition, the problem would be far less obvious and the results could be disastrous.

Now explicitly execute the modified startup command file in this cell to force the new environment setting to be used.

3:**@START**[14]

[14] Remember that to change to the third top-level menu you press the right mouse button or space bar twice.

Now the box on M2 should be colored brown. The new technology settings have been read from the startup command file and stored in the environment of this cell.

Now let us try to save the cell with the usual method.

1:**FILE → LEAVE**

Reopen the cell to verify that the new environment setting was saved in the cell file.

ICTRIV NEWCELL

Oh no! The box is white again! If you followed the instructions as listed, the cell file was not saved after you changed the environment settings. This is because of the way the LEAVE command works. The LEAVE command is usually the best command to use to close the layout editor since it saves only cells where you have changed the geometry. However in this case you changed only an environment setting, so the cell file was not saved. **If you want to be sure of saving changes to a cell's environment, use the EXIT command instead of the LEAVE command.**

Now re-execute the startup command file and use the EXIT command to force the editor to save the cell file.

3:**@START**

1:**FILE → EXIT**

Open the editor one more time to verify that the M2 layer is now consistent with the modified technology setting.

ICTRIV NEWCELL

Now exit the editor again.

1:**FILE → QUIT**

You can see that changing a technology setting after cells are created can cause headaches. Avoid this type of problem, and the far more dangerous types of discrepancies caused by other changes to technology settings that may go undetected, by trying to avoid changing startup command files mid-design.

If you must change a critical technology setting after some cells are created, you must insure that the old technology settings do not remain in any existing cells. One way to do this is to use the STREAM command to export all cells to stream format, then use the UNSTREAM utility to re-import them to cell file format. This will strip the environment from all cell files. When you edit any of these cells, the modified startup command file will be executed automatically. Another method to replace technology settings will be covered in later tutorial on page 254.)

A Bipolar Multiple Project Example

Dick is working on 2 projects simultaneously using a 1.5 micron bipolar technology: a bipolar operational amplifier using a standard cell library, and a phase-locked loop using the same standard cell library and process.

Dick is working alone on these designs. He does not need to share files with other users, but he should use the same technology file in both bipolar projects. He will also have some common cell libraries in the two projects.

Q:\ICWIN\	installation directory for the ICED™ software
bipol.bat	new project batch file for both projects
TECH\	main directory for technology dependent files
BI1.5\	new subdirectory for the 1.5µ bipolar-specific command files
bipolnew.cmd	new startup command file for this process
TUTOR\	parent directory for tutorial cell files
OPAMPWORK\	working directory for opamp project
PLLWORK\	working directory for phase-locked loop project
BI1.5_CELLS\	Parent directory for bipolar cell libraries
SHARED_CELLS\	Customized cells shared by both projects
STD_CELLS\	Standard cells shared by both projects

Figure 23: Files and directories used in the bipolar example. Directory names are in upper case. File names are in lower case. The names of objects created in this tutorial are shown in **bold** characters.

Creating the Startup Command File for the Bipolar Projects

The 1.5-micron bipolar processes will be used for two different projects. However a single startup command file can be shared by both projects. It should be located in a new subdirectory of the Q:\ICWIN\TECH directory.

Remember that it is best to store all support files related to a specific technology together in a technology directory. You should have a separate technology directory for each process.

Use whatever method you are comfortable with to create the new subdirectory, Q:\ICWIN\TECH\BI1.5. Then copy the sample startup command file supplied with the installation to a new file BIPOLNEW.CMD in this new technology directory. (See page 58 for sample DOS commands to accomplish this.)

Now edit this file (Q:\ICWIN\TECH\BI1.5\BIPOLNEW.CMD) with your favorite text editor to customize it for this process. The Bipolar process has different layer definitions than the sample process. Change the LAYER lines in the file as follows:

Old: LAYER 1 NAME=NWEL WIDTH=3.000 ….
New: LAYER 1 NAME=**N** WIDTH=3.000 ….

Old: LAYER 2 NAME=NDIF WIDTH=3.000 ….
New: LAYER 2 NAME=**N_PLUS** WIDTH=**1.500** ….

Old: LAYER 3 NAME=PDIF WIDTH=3.000 ….
New: LAYER 3 NAME=**P** WIDTH=3.000 ….

Other changes to the layer definitions may also be required in a real bipolar startup command file, including changes to patterns and colors. But these changes to the layer definitions are all we will make here. We will make one more change to this file. Add the following line as the very last statement in the file.

New: **$ %start.cmd executed successfully**

After a command file is executed, the last statement in the command file remains on the history line of the editor. The %start.cmd parameter is a system macro that contains the name of the startup command file. When this startup command file is executed, the line above will cause a comment reporting the name of the startup command file on the history line. (To learn more about system macros, see the Creating Useful Command Files tutorial and the ICED™ Command File Programmer's Reference Manual.)

Save the file now and close the text editor.

The Cell Library Search Path for the Bipolar Projects

Each project should have it's own working directory. Create the following directories:

Q:\ICWIN\TUTOR\OPAMPWORK
Q:\ICWIN\TUTOR\PLLWORK

The Bipolar standard cell library will be shared by both projects. We also need to create a library of reusable cells from older bipolar projects. Create the following directories:

Q:\ICWIN\TUTOR\BI1.5_CELLS\STD_CELLS

Q:\ICWIN\TUTOR\BI1.5_CELLS\SHARED_CELLS

The cell search path for the opamp project will be as shown below. We will apply **copy-edit** protection to the SHARED_CELLS library so that Dick can save customized copies of these cells in the working directory of each project. He can edit cells in a copy-edit library, but modified cells will be saved to his current working directory. Since the working directory is always first in the cell search path, these customized copies of cells will be used by the layout editor, and the original copies in the SHARED_CELLS library will be ignored.

He will be prevented from making any changes to cells in the STD_CELLS directory since we will apply **view-only** protection to this cell library. Each working directory is a **direct-edit library**. He can edit cells in these libraries and save them again in the same directory.

Search direction ↓	**Cell Library**	**Method of Protection**
	Q:\ICWIN\TUTOR\OPAMPWORK	Direct-edit
	Q:\ICWIN\TUTOR\BI1.5_CELLS\SHARED_CELLS	Copy-edit
	Q:\ICWIN\TUTOR\BI1.5_CELLS\STD_CELLS	View-only

Figure 24: Cell search path for opamp project

The search path for the Phase-Locked Loop project is almost identical. The only change from the Opamp project is the name of the working directory.

Search direction ↓	**Cell Library**	**Method of Protection**
	Q:\ICWIN\TUTOR\PLLWORK	Direct-edit
	Q:\ICWIN\TUTOR\BI1.5_CELLS\SHARED_CELLS	Copy-edit
	Q:\ICWIN\TUTOR\BI1.5_CELLS\STD_CELLS	View-only

Figure 25: Cell search path for phase-locked loop project

Creating the Bipolar Batch File

Since the only difference between the search paths is the working directory, and the startup command file is the same for both projects, they can share a batch file for opening the editor.

Make a copy of the ICWIN.BAT file supplied with the installation. Use the file name BIPOL.BAT. Remember that all layout editor batch files should be stored in the Q:\ICWIN directory. Now edit the Q:\ICWIN\BIPOL.BAT file to modify the cell library search path.

Protection method	**ICED_PATH switch**
View-only	/v (default)
Copy-edit	/c
Direct-edit	/d

Figure 26

Remember that cell library directory names and methods of protection are stored in a system environment variable with the name ICED_PATH. The protection method for each cell library is set with a single character following each directory name. The protection character is separated from the directory name with a '/'. The default protection method used for a library (when no protection character is used in the ICED_PATH definition) is view-only.

Old: set iced_path=q:\icwin\samples;

New: set iced_path=q:\icwin\tutor\bi1.5_cells\shared_cells /c +

The '+'[15] at the end of the line allows you to continue the cell library definition in environment variables with the names ICED_PATH_1, ICED_PATH_2, etc. Define the other cell library with default view-only protection with the following change:

Old: set iced_path_1=

New: set iced_path_1=**q:\icwin\tutor\bi1.5_cells\std_cells**

The working directory definition is not included in this batch file.

The next line in the batch file defines the ICED_CMD_PATH environment variable. This variable stores the search path for command files. This time we will assume that Dick does have some specialized command files for this technology, so we should change this line to force the editor to search this directory first for command files.

Old: set iced_cmd_path=q:\icwin\tech\samples;

New: set iced_cmd_path=q:\icwin\tech\bi1.5;

[15] In older versions of ICED™, the '&' was used as a continuation character in batch files. This is no longer recommended.

The last change is to replace the reference to the startup command file. We will refer to the file created back on page 64.

Old: q:\icwin\icwin %1 start=q:\icwin\tech\samples\new menu=* …

New: q:\icwin\icwin %1 start=q:\icwin\tech\bi1.5\bipolnew.cmd menu=* …

Our changes are now complete. **Save the file** and exit the text editor.

Opening the Editor for the Bipolar Projects

When Dick needs to open the editor, he first opens the console window with the ICED icon on the desktop. He must change to the working directory of the appropriate project before executing the batch file to open the editor. To create a new cell “NEWCELLB” in the opamp project, type:

CD TUTOR\OPAMPWORK

BIPOL NEWCELLB[16]

Even though the batch file BIPOL.BAT is stored in the Q:\ICWIN directory, rather than the current directory Q:\ICWIN\OPAMPWORK, the system will be able to find the batch file since the ICED icon adds the Q:\ICWIN directory to the system executable search path.

To work on the phase-locked loop project, Dick will first need to make the Q:\ICWIN\TUTOR\PLLWORK directory the current directory. Then he executes the same batch file. New cells, and modified copies from the SHARED_CELLS library, will be saved in this working directory. Different copies of cells with the same name may wind up in each of the working directories. This is acceptable because each project’s search path will not include the working directory of the other project.

[16] If you get a message from the operating system about a lack of environment space while executing this batch file, you need to increase the amount of memory available for environment variables. If you do not know how to do this (the method varies depending on the operating system,) contact IC Editors technical support.

A Shared Project on a Network

Dick has a co-worker named Jane. He and Jane will collaborate on a microprocessor design using a .5µ CMOS process. They need to share technology files, project cell libraries, and a standard cell library for this project. The shared files will be located on a networked computer.

We will use the drive letter N to represent the drive of the networked computer. Dick's drive will be represented by the drive letter D. J will be used as Jane's drive letter. We will assume that both Dick and Jane have the ICED™ program installed on their own computer, but this example would be almost identical if they both used a copy of the program installed on the networked computer.

N:\ICWIN\	base of the directory tree on the networked computer for ICED™ files
TECH\	main directory for technology dependent files
CMOS.5\	new subdirectory for the shared .5µ CMOS process-specific command files
cmos5new.cmd	new startup command file for the .5µ CMOS technology
PROCESSOR\	base of the directory tree for shared project cell libraries
DICK_CELLS\	shared cell library where Dick stores completed cells
JANE_CELLS\	shared cell library where Jane stores completed cells
CMOS.5_CELLS\	base of the directory tree for cell libraries that may be used by other .5µ CMOS projects
SHARED_CELLS\	reusable cells that may need future customization for this project
STD_CELLS\	standard cells that should not be altered
D:\\ICWIN\\	Dick's installation directory
proc.bat	Dick's project batch file
PROCESSOR\	Dick's working directory for this project
J:\\ICWIN\\	Jane's installation directory
proc.bat	Jane's project batch file
PROCESSOR\	Jane's working directory for this project

Figure 27: Files and directories used in the network example

Creating the Startup Command File for the CMOS Project

First, a new technology directory for this process must be created. Dick and Jane will both need access to this CMOS technology directory, so we will assume that it is located on a shared network drive. We will call this network drive "N:". You can locate this directory on any drive to complete the tutorial, but if you have access to a shared drive, use that drive letter to replace the "N" drive letter in the examples. If you have access to only a single drive, just use that drive for the tutorial.

You can use whatever method you are familiar with to create the technology directory, N:\ICWIN\TECH\CMOS.5. We list the DOS commands to do this below. Type the following in the console window to create the directory and copy the sample startup command file:

```
N:[17]
CD \
MD ICWIN
CD ICWIN
MD TECH
CD TECH
MD CMOS.5
CD CMOS.5
COPY Q:\ICWIN\TECH\SAMPLES\NEW.CMD  CMOS5NEW.CMD
```

Now edit this file (N:\ICWIN\TECH\CMOS.5\CMOS5NEW.CMD) with your favorite text editor to customize it for the CMOS process.

The RESOLUTION command in the file sets the finest step size of the grid used when digitizing coordinates. The mouse cannot be used to digitize a coordinate that is not on the resolution grid. The setting in the sample startup command file is 0.500. This is probably too coarse a grid for a .5 micron technology. To change the grid to a .1 step (allowing you to digitize the coordinate (10.8, 11.3) for example), change the following line in the file:

```
Old:    RESOLUTION  STEP=0.500  MODE=SOFT
New:    RESOLUTION  STEP=0.100  MODE=SOFT
```

The SNAP command allows you to set a more restrictive grid for snapping coordinates digitized with the mouse. Change this grid to a .1 step also.

[17] Remember that N: represents a shared drive. Replace the "N" with an appropriate drive letter.

Old: SNAP ANGLE=45 STEP=(0.500,0.500) OFFSET=(0.000,0.000)
New: SNAP ANGLE=45 STEP=(0.**1**00,0.**1**00) OFFSET=(0.000,0.000)

The finest display grid should also change. Change the grid that will be displayed on the screen so that the finest grid displays dots every .5 units by changing the following line:

Old: GRID 1 ON COLOR=RED DOTS STEP=1.0
New: GRID 1 ON COLOR=RED DOTS STEP=**0.5**

The next change to this startup command file is to change the default wire widths of the M1, M2, and POLY layers to more appropriate values for a .5 micron technology. Make the following changes to the file:

Old: LAYER 5 NAME=POLY WIDTH=2.000 ….
New: LAYER 5 NAME=POLY WIDTH=**0.500** ….

Old: LAYER 6 NAME=M1 WIDTH=3.000 ….
New: LAYER 6 NAME=M1 WIDTH=**1.500** ….

Old: LAYER 7 NAME=M2 WIDTH=4.000 ….
New: LAYER 7 NAME=M2 WIDTH=**2.000** ….

One more change that may be helpful for this technology is to change the default display depth. Let us say that Dick and Jane usually prefer to open the editor to see the entire chip, then zoom in on the area of interest to do their work. Displaying all cells of such a dense design will result in a noticeable delay each time they open the editor. If only the first few levels of cells are displayed instead, the display draws itself much faster, usually with 0 noticeable delay.

The DISPLAY command is used to control several aspects of how the editor displays a design. The CELL_DEPTH parameter of this command controls how many cell levels will be displayed. A value of 3 for this parameter means that only 3 levels of subcells will be displayed. Edit the line as follows to add this parameter to the line beginning with the keyword "DISPLAY" near the top of the file:

Old: DISPLAY CELL_LABELS=ON ….
New: DISPLAY **CELL_DEPTH=3** CELL_LABELS=ON ….

(Note that a line that includes a CELL_DEPTH keyword is included earlier in the file. However this line begins with a '$'. It is merely a comment line. The TEMPLATE command that created this NEW.CMD file comments out the CELL_DEPTH and MENU settings for a reason. Both of these settings can be overridden on the command line in the batch file used to open the editor. If a startup command file that contains a CELL_DEPTH or MENU setting is then executed, it will override the settings on the command line. So the default creation of a startup command file is to add these settings to the file only as comments, then the command line options will never be overridden. However, you can use an uncommented DISPLAY CELL_DEPTH command in the startup command file explicitly as we have done above. Just keep in mind that it will be impossible to override the CELL_DEPTH on the command line with the DEPTH parameter when opening a new cell file.)

The settings in the startup command file are only defaults for new cells. Once the editor is open, this DISPLAY parameter (or any parameter defined in the startup command file) can be changed by the user. The changed value is stored with the cell file. For this type of environment setting, it is acceptable (and even beneficial) to have different values in different cell files.

The last change to this file is to add the following line as the very last statement in the file.

New: **$ %start.cmd executed successfully**

There may be other changes to this file that would be helpful for Dick and Jane. They should agree on layer colors and patterns for display and plotting. They may want to modify, or add to, the keyboard shortcuts defined at the end of this file. However, we won't make any more changes to this file. Save the file now and close the text editor.

The Cell Library Search Path for the CMOS Project

Both of our designers should have a separate working directory where new cell files are stored. A working directory should always be stored on the user's local drive. Dick and Jane should each create the directory \ICWIN\PROCESSOR on their own drive.

The standard cells for this project should be stored on the networked computer so that both Dick and Jane always use the same cells. Create the directory N:\ICWIN-\CMOS.5_CELLS\STD_CELLS to store the standard cells from the foundry.

Assume that Dick and Jane have cells from previous projects in this technology that they intend to reuse in this microprocessor. Create a directory to store these reusable cells, N:\ICWIN\CMOS.5_CELLS\SHARED_CELLS.

As Dick and Jane complete cells that they need to share for this project, they should store them on the network so that they are using identical copies of the cells. However, Dick would prefer that Jane only have read access to the directory where he stores his cells, so that she cannot accidentally modify any of his cells. Jane feels the same way about her work. So create two other directories, N:\ICWIN\-PROCESSOR\DICK_CELLS and N:\ICWIN\PROCESSOR\JANE_CELLS.

The cell search path Dick should use to search for subcell files should be as shown on the next page. He should only be allowed to read the cells in most of these libraries. Using them as working directories may corrupt Jane's work. He should be able to make modifications to the cells in the N:\ICWIN\PROCESSOR-\DICK_CELLS directory if required. However, he does not want to use the DICK_CELLS library as a working directory since it is on a networked computer and constantly updating the log file over the network will slow down his edit session. Also, he may want to work on a cell in this library for a few days before having Jane access the new version. So this cell library should be protected as a copy-edit library. He can edit cells in the library, but changed cells will be saved to his working directory. His working directory is a direct-edit library. He can edit cells in this library and save them again in the same directory.

	Cell Library	**Method of Protection**
Search direction ↓	D:\ICWIN\PROCESSOR	Direct-edit
	N:\ICWIN\PROCESSOR\DICK_CELLS	Copy-edit
	N:\ICWIN\PROCESSOR\JANE_CELLS	View-only
	N:\ICWIN\CMOS.5_CELLS\SHARED_CELLS	View-only
	N:\ICWIN\CMOS.5_CELLS\STD_CELLS	View-only

Figure 28: Dick's cell library search path

The cell libraries will be searched in the order shown. If Dick edits a cell in the DICK_CELLS library, the modified copy will be saved in his PROCESSOR directory. The next time the editor searches for this cell, it will use the copy in his PROCESSOR library, and the copy in the DICK_CELLS library will be ignored. He must remember to move the finished cell back to the DICK_CELLS library when the work is complete, so that Jane has read access to the current version of the cell. Dick will not be allowed to edit any of the cells in the JANE_CELLS library, even if he forgets who owns what.

Jane's search path is very similar, but the library with her work is first in the search path with copy-edit protection.

Search direction ↓	Cell Library	Method of Protection
	J:\ICWIN\PROCESSOR	Direct-edit
	N:\ICWIN\PROCESSOR\JANE_CELLS	Copy-edit
	N:\ICWIN\PROCESSOR\DICK_CELLS	View-only
	N:\ICWIN\CMOS.5_CELLS\SHARED_CELLS	View-only
	N:\ICWIN\CMOS.5_CELLS\STD_CELLS	View-only

Figure 29: Jane's cell library search path

Two cells with the same name should never exist in the DICK_CELLS and JANE_CELLS libraries. This would lead to different versions of completed cells being used by Dick and Jane. They must be careful to communicate enough to not use an existing name for a new cell.

If Dick decides that a cell in the view-only library SHARED_CELLS needs to be modified, then he should **move** (not copy) the cell file to the DICK_CELLS directory first. This will allow Jane to still access the cell file, but she will be unable to modify it. It has been "checked out" by Dick allowing only him to modify it. He then saves a copy in his working directory while he works on it. When he is finished, he **moves** it back into the DICK_CELLS directory. This gives Jane access to the current version, while reserving Dick's right to modify it again in the future.

Creating the CMOS Batch File

Now make a copy of the batch file supplied with the installation (Q:\ICWIN\ICWIN.BAT) and customize it as the batch file Dick would use to open the layout editor for the CMOS project. Copy the file to D:\ICWIN\PROC.BAT. (Always store these batch files in the Q:\ICWIN directory. We strongly recommend that you never copy batch files into your working directories or any other locations. If you modify one copy, you may wind up executing an old copy somewhere else without realizing it, and the cell search path is then mysteriously incorrect. The Q:\ICWIN directory is always the first directory in the system executable search path when you use the console window opened by the ICED icon.)

Now edit the D:\ICWIN\PROC.BAT file to modify the cell library search path. The first library on the search is always the working directory, or current directory. The working directory is always a direct-edit library. You should not specify this directory in the ICED_PATH variable. You can change to this directory with DOS commands before you execute the batch file, but you can add the command to change to this directory in the batch file itself. Add the following lines to the PROC.BAT file directly after the first line. The first line below changes the current drive; the second changes the current directory.

New:	**D:**
	CD \ICWIN\PROCESSOR

This batch file is now no longer general purpose. It can be used only to open cells in this working directory. This makes it easier to open the editor for cells in this project, since you do not need to explicitly change to this directory before executing the batch file. However, you cannot use this batch file to open a cell in a different directory as the root cell.

Edit PROC.BAT to change the cell libraries as shown below.

Old:	set iced_path=q:\icwin\samples;	
New:	set iced_path=**n:\icwin\processor\dick_cells/c**	+
Old:	set iced_path_1=	
New:	set iced_path_1=**n:\icwin\processor\jane_cells**	+
	set iced_path_2= n:\icwin\cmos.5_cells\shared_cells	+
	set iced_path_3= n:\icwin\cmos.5_cells\std_cells	

Alternately, you can define all cell libraries on the same ICED_PATH line, as long as they are separated with semicolons, and the total of all characters on the line does not exceed the operating system limit on command line length (rarely a problem.)

The next line in the batch file defines the environment variable that contains the search path for command files, ICED_CMD_PATH. The batch file supplied with the installation sets this variable to Q:\ICWIN\TECH\SAMPLES. Let us assume that Dick and Jane have no unique command files other than the startup command file. In this case, the search path for command files needs no alteration. The path to the startup command file will be defined in the command line used to execute the layout editor.

The DRC_PATH line needs no alteration unless the DRC (Design Rules Checker) tool will be used.

We need to modify the layout editor command line to replace the reference to the startup command file. We will refer to the file created back on page 69.

Old: q:\icwin\icwin %1 start=q:\icwin\tech\samples\new menu=* …

New: q:\icwin\icwin %1 start=**n:\icwin\tech\cmos.5\cmos5new.cmd** menu=* …

Our changes are now complete. **Save the file** and exit the text editor.

Jane's project batch file will be very similar. If you have access to another computer on the network, copy the PROC.BAT file created above to J:\ICWIN\PROC.BAT and edit it as shown on the next page.

The first change is to use Jane's drive letter in the first line that changes the current drive in the batch file.

Old:	D:
New:	**J:**

Also change the search order of the cell libraries and set the permissions correctly for the search order as shown in Figure 29 on page 73.

Old:	set iced_path=n:\icwin\processor\dick_cells/c	+
New:	set iced_path=n:\icwin\processor\jane_cells/c	+
Old:	set iced_path_1= n:\icwin\processor\jane_cells	+
New:	set iced_path_1=n:\icwin\processor\dick_cells	+

The changes are complete. Save the file and exit the text editor.

Opening the Editor for the CMOS Project

Now that the files are set up, opening the editor each time is very simple. Open a DOS console window with the ICED icon on the desktop. To create a new cell "PROCCELL" in Dick's D:\ICWIN\PROCESSOR directory, Dick would type at the console prompt:

PROC PROCCELL[18]

[18] If you get a message from the operating system about a lack of environment space while executing this batch file, you need to increase the amount of memory available for environment variables. If you do not know how to do this (the method varies depending on the operating system,) contact IC Editors technical support.

The D:\ICWIN\PROC.BAT file will be executed. The command to change to the D:\ICWIN\PROCESSOR working directory is included in the batch file, so he does not even need to make this the current directory first. The environment definitions in the startup command file are executed automatically. The editor is now open and ready for work in the new cell. Create some geometry if you like, then close the editor.

Eventually one of the designers will have to "own" the main cell for the entire chip. Let us assume that Jane is assigned this cell. We will call it PROCCHIP. Once she has a first version of this cell available, she moves the cell file to the N:\ICWIN\JANE_CELLS library.

Suppose that Dick wants to open this main cell to work on his subcells within it. He should not use JANE_CELLS as his working directory. There is an easy method around this problem. He creates a "dummy" cell in his own working directory and launches the editor to open this cell file. He can then add the PROCCHIP cell to this dummy cell, or simply use the EDIT command in the editor to open the cell in view-only mode. He will be able to access the cells he "owns" for editing even though the PROCCHIP cell cannot be modified. Using this method there is only one copy of this main cell and no risk of the different designers both modifying the same file so that their versions of the main cell get out of sync.

Conclusion

This concludes the tutorial. You can experiment with adding other features to the startup command files or the batch files. Any editor command can be added to the startup command file. Refer to the ICED™ Layout Editor Command and Utility Reference Manual to learn more about available commands. Some other customizations you may want to make are shown in Figure 30 below.

If you want to add more advanced features to your startup command file, you should read the Command File Programmers Reference Manual. Remember that it is best to finalize the startup command file **before** creating any cells for a new project. Changes made to the file after cells are created will not be reflected automatically in existing cells.

You can add any valid DOS commands to the batch file. If you want to learn more about the options you can add to the layout editor command line, see the Command Line Parameters Reference in the ICED™ Layout Editor Reference Manual.

Parameter to change in startup command file	**Purpose**	**Refer to page in manual**
LAYER command STREAM parameter	Define the layer number used for stream format export	214
LAYER command SPACER parameter	Define minimum space between shapes on a layer for appearance of spacing cursor	121
LAYER command PAT parameter	Define the fill pattern for display	173
LAYER command PEN parameter	Define the fill pattern for plots	191
GRID *n* command	Define display grids	185
RESOLUTION command	Define the minimum resolution for vertex points	209
SNAP command	Define a temporary, more restrictive, grid for snapping vertices	123
COLOR command ONE_DOT and FOUR_DOTS parameters	Define color used in plots	194
GLOBAL KEY.F*n* commands	Define keyboard shortcuts	231

Figure 30: A few of the important environment settings in the startup command file.

Removing Files Created by This Tutorial

We created many files in this tutorial, so you may want to clean up after yourself and remove all of these files from the affected drives. Alternately, you may want to leave these files around for reference.

If you do want to remove them, you can use the DOS commands on the following page to remove the files, or use whatever method you are comfortable with.

Files on your computer's drive

```
Q:
CD \ICWIN
DEL ICTRIV.BAT
DEL BIPOL.BAT
DEL PROC.BAT
CD TECH
DEL TRIVIAL
DEL BI1.5
DEL CMOS.5
CD \ICWIN
DEL PROCESSOR
CD TUTOR
DEL *.*
```
[19]

Files on the networked computer's drive

The delete procedure below assumes that you have no real ICED™ files on the networked computer. If so, you can delete the entire directory. If not, do not execute the DEL *.* below, instead delete only the files shown in Figure 27 on page 68.

```
N:
CD \ICWIN
DEL *.*
```
[19]

[19] Always insure that you are in the directory you think you are before executing DEL *.*. If you mistype the CD command above, and are still in the root directory of the drive, the delete command could have disastrous results.

More on Entering Commands

We have already covered the basics of entering commands. This tutorial goes into more depth on how to select ICED™ operations.

Commands can be entered in 4 ways:

- Selecting commands from the menus
- Typing on the keyboard:
  - Whole commands on the command line
  - Single function keys that have been assigned a command macro
- Executing command files

All functions available on the menus can also be executed by typing commands at the command prompt[20]. Typing the command at the prompt often allows you to use more options than the menus offer. The menu item name is often a shortened variation on the command name.

Use the ICED icon and the ICWIN.BAT file to open the editor to edit any cell. You can use the same cell you created in the Basics Tutorial, or you can open the editor to edit a new cell. If you need to review how to do this, follow the directions in the Basic Tutorial on page 17.

Review of Menus

Remember that there are several top-level menus. You cycle through these menus with the right mouse button or the <Space bar >.

Many menu entries have sub-menus that are displayed when the menu item is selected. For example, the VIEW menu entry on the first top-level menu opens a sub-menu of options to change the view window. Look at this sub-menu now by selecting the following menu item with the left mouse button:

1:[21]**VIEW**

[20] The 1:Again menu option is the only exception to this statement. There is no exact equivalent in a typed command.
[21] Remember that the number before the colon indicates the number of the top-level menu.

Part way down the sub-menu is the entry “LAST”. This option changes the view window to the previous view. We use the following notation to indicate this menu entry 1:**VIEW → LAST**. Click this option now. Note that the actual command that was executed, “VIEW LAST”, is reflected on the history line below the command prompt.

The most frequently used options of the VIEW command are also repeated under the VIEW heading of the first top-level menu as a convenience. Click on the item “last” under the heading “VIEW” on the first top-level menu. (We use the notation 1:(VIEW)**last** to refer to this menu option.) Note on the history line that the same command, “VIEW LAST”, was executed again. This menu entry performs the exact same operation as the 1:**VIEW → LAST** menu option. You can select view options here, or from the complete list by selecting 1:**VIEW** first.

Now add a component to the drawing by selecting the following menu entry:

1:(ADD)**box**

Note that the menu disappears entirely while you are executing a command. The cursor can now only be used to digitize corners of the new box. Digitize two corners of the box by moving the cursor in the view window and pressing the left mouse button once for each corner.

The menu entries sometimes change to be even more convenient. For example, when nothing is selected, the UNSELECT option is not displayed on the first top-level menu. However, once a component is selected, the menu will change. The entries under ADD are replaced with several entries under the UNSEL heading that are useful for unselecting components. Note these changes to the menu after you select the new box with the following menu entry:

1:(SELECT)**new**

The 1:(ADD)**box** entry is now no longer present on the first top-level menu. So to add another box, you must use the menu entry:

1:**ADD → box**

When you select commands from the menu, note that the editor command generated from your choices is echoed on the command line.

The following table lists each major menu entry in the menu supplied with the installation in the order it appears on the menu, locations in this manual where the menu entry is used in examples, and the name of the command generated. You would use the name of the command to look for more details on the operation in the Layout Editor Reference Manual.

Menu entry	**Command(s) generated**	**Page**
1:**DELETE**	DELETE	33
1:**VIEW**	VIEW	22
1:**VIEW → LIMIT**	VIEW LIMIT	182
1:**COPY**	COPY	26
1:**MOVE**	MOVE	28
1:**UNDO**	UNDO	24
1:**Again**	(no equivalent command)	24
1:**UseLay**	USE LAYER ADD	23
1:**SELECT**	SELECT	96
1:**ADD**	ADD	21
1:**UNSEL**	UNSELECT (refer to SELECT command)	45
1:**USE**	USE	113
1:**FILE →@START**	*@file_name* (executes startup command file)	61
edit.JOU	(no equivalent command)	225
MENU	MENU	-
PATTERN	PATTERN	174
DRC *xxx*	DRC	-
CIF	CIF	-
STREAM	STREAM MAP_LAYERS	214
ARCHIVE	STREAM ARCHIVE	216
PLOT	PLOT (create plot file)	201
(PLOT)**now**	PLOT (create plot file and print)	190
LEAVE	LEAVE	47
EXIT	EXIT	47
QUIT	QUIT	160
JOURNAL	JOURNAL	29
PACK	PACK	-
2:**SWAP**	SWAP	38
2:**TEMPLA**	TEMPLATE	33
2:**SHOW**	SHOW	35
2:(SHOW)**@one**	(no equivalent command, Displays information on selected components one at a time, leaving last component selected)	106
2:(SHOW)**@deep**	@DEEPSHOW.CMD	157

(Continued on next page.)

2:**RULER**	RULER	31
2:**EDIT →EDIT**	EDIT	163
P_EDIT	P_EDIT	160
T_EDIT	T_EDIT	159
GROUP	GROUP	46
UnGRP	UNGROUP (refer to GROUP command)	44
2:**MERGE**	MERGE	39
2:**CUT**	CUT	41
3:**ReDraw**	REDRAW	-
3:**DOS**	DOS (open console window or execute DOS command and wait)	90
3:**SPAWN**	SPAWN (open console window or execute DOS command without waiting)	215
3:**@START**	*@file_name* (executes startup command file)	61
3:**@%.cmd**	*@file_name* (lists all available command files)	172
3:**PROTEC**	PROTECT	102
3:**UNPROT**	UNPROTECT (refer to PROTECT command)	102
3:**BLANK**	BLANK	176
3:**UNBLNK**	UNBLANK (refer to BLANK command)	176
3:**LAYER**	LAYER	114
3:**FILL**	FILL	40
3:**OUTLIN**	OUTLINE	178
3:**A-PAN**	AUTOPAN	179
3:**GRID**	GRID	185
3:**SNAP**	SNAP	123
3:**RESOLV**	RESOLUTION	149
3:**NEAR**	NEAR	-
3:**DISPLY**	DISPLAY	183
3:**ARRAY**	ARRAY (controls display of arrays, not creation)	184
3:**TEXT**	TEXT (controls display of text, not creation)	126
3:**ARROW**	ARROW	179
3:**SPACER**	SPACER	121

Figure 31: ICED menu options and the commands generated by them

Custom Menus

There are ways to make the menus even more convenient. You can rearrange the menus to place the entries you use most frequently on the top-level page. You can also add menu entries. This is particularly useful when you have learned how to use command files. You can add menu entries to execute your favorite command files.

The MkMenu utility is used to create customized menus. However, this is a subject too advanced for us to cover here. If you want to customize menus, contact technical support at IC Editors, Inc.

Uses of the Mouse

We have already covered the basic uses of the mouse in the Basics Tutorial. The table below is a complete list of uses of the mouse. Experiment with each of the mouse button uses listed until you feel comfortable with all uses of the mouse.

Button	Current Operation	Use
Left button	None (menu displayed)	Selects menu entries
	Digitizing coordinates	Defines point at current location
Right button	None (menu displayed)	Cycles through top-level menus
	Selecting items from a list	Indicates that selection is complete
	Digitizing coordinates for a wire, line, or polygon	Indicates that you have finished entering coordinates
	Embedded selection for MOVE or COPY	Allows you to cancel first point for displacement vector and redigitize it
	Executing other layout commands	Moves the cursor to the center of the view window
Both buttons simultaneously	Any operation	Cancels menu selection or command and returns to first menu
Center button on a three button mouse (or <Esc> key)	None (menu displayed)	Cancels menu selection and returns to first top-level menu
	Any edit command	Invokes the nested view menu

Figure 32: Uses of the mouse buttons

Typing Commands at the Command Line

Commands can be entered by typing on the keyboard any time a menu is displayed. The cursor is automatically active on the command line except when a command is currently being executed. You do not need to move the cursor or switch away from the view window.

Importance of Correct Syntax

When you use the keyboard to enter a command, you must type all required keywords of the command. For example, try to enter an ADD command by simply typing the following at the keyboard:

ADD <Enter>

Since this is not a complete ADD command, the program will inform you of a syntax error below the history line of the window. The history line will echo the command as you have typed it and it will indicate the missing or incorrect parameter with chevrons, "ADD <<>>". The syntax warning reads "error: expected component type keyword". This means that the program expected a component type keyword in place of the chevrons.

The component type keyword is a required parameter of the ADD command. You can refer to the correct syntax of this command, or any command, in the ICED™ Layout Editor Reference Manual.

Another way of learning the correct syntax for a command is to use the menu to select an operation, then use the command history retrieval method (covered below) to see the exact syntax of the command generated by the menu selections.

Correcting Mistyped Commands

You can correct the syntax of the ADD command by retrieving the command from the command history list. Press the < ↑ > key once. (If this keystroke does not retrieve the last command, try holding the <Ctrl> down while pressing the < ↑ > key. The mode of the arrow keys is controlled by ARROW command. The ARROW mode may be set to the non-default mode in your cell.)

The partial ADD command should now be present on the command line with the cursor at the end of the line. Type the following:

<Space bar> BOX <Enter>

Now that the entire command "ADD BOX" has been entered, the editor will execute the command and wait for you to digitize the corners of the new box. Input from the keyboard is ignored while the command is executing[22]. You can cancel the command by pressing both mouse buttons at the same time. Do so now. This returns you to the first top-level menu and frees the keyboard so that you can type on the command line again.

From now on, we will not include the <Enter> at the end of each sample command line. Simply type the text as shown on the command line, and then press <Enter>.

You can use the following keys to move the cursor (i.e. the text insertion point) on the command line to correct mistyped commands:

<Home>, <End>, < ← >, and < → > keys

Press the < ↑ > key once to retrieve the ADD BOX command again. Now use the keys above to move the cursor around on the command line. Then, use the <Home> key to place the cursor at the beginning of the ADD BOX command line.

The command line is usually in insert mode. You can switch to overtype mode, where you type over text already on the command line by typing the <Insert> key on the keyboard once. Type over the text on the command line to see how this works. Type the <Insert> key again to toggle the keyboard back to insert mode and type a few more characters to test this mode. Press both mouse buttons to clear the command line.

Keyword Abbreviations

Most of the keywords in ICED™ commands can be abbreviated to the first few letters of the keyword. You need to type only enough letters of each keyword to make it unambiguous. For example, the ADD BOX command can be typed as:

ADD B

AD BOX

or even

AD B

The shortest valid abbreviation for the ADD keyword is "AD" since other commands begin with the letter 'A'. However, no other component types begin with the letter 'B', so the single letter abbreviation "B" is sufficient to be unambiguous.

[22] Input from the keyboard is ignored except for the <Esc> key that allows you to change the view window while digitizing the necessary coordinates.

When you have not typed enough letters to be unambiguous, ICED™ will warn you. Type the following now to see the warning:

A B

Assigning Commands to Function Keys

You can easily assign entire commands to the function keys <F1> - <F12>. The startup command file executed when you created this cell probably assigned a few for convenience. Let us see what command was assigned to the <F1> key. Press it now.

You are probably executing the RULER command now. Press both mouse buttons to cancel it if you are.

You can reassign <F1> to the SHOW command (used to display information about selected components) by typing the following command at the command line:

KEY F1 = SHOW SELECT FILE=*

Now press <F1> again. The text on the command line indicates that you are indeed executing the SHOW command. ICED™ is waiting for you to select a component so it can display the component information. Press both mouse buttons to cancel.

This key assignment is stored in the cell definition. When you save the cell file, it will be saved in the file. The next time you edit the same cell, you can press <F1> and it will execute the SHOW command.

However, **other cell files will remain unaffected by this assignment**. You need to make the key assignments in the startup command file if you want the same key assignments in every new cell file.

ICED™ does not limit you to defining only one command string to each of the function keys. You can also use the KEY command to assign commands to combinations of the function key and the <Alt>, <Shift>, or <Ctrl> keys. To assign the RULER command to the key combination <Alt><F1>, type the following command:

KEY AF1 = RULER

Now press the <Alt> and <F1> at the same time to see what happens. Then cancel the command with both mouse buttons.

Assigning Commands to Ordinary Keys

The KEY command you used above is merely shorthand for defining the most common type of keyboard macros, those assigning command strings to function keys. You can use another form of keyboard macro definition that is less restricted and more powerful. This method of keyboard macro definition allows you to assign a command string to any combination of ordinary keys.

The syntax of the more general-purpose keyboard macro definition is as follows:

GLOBAL #KEY.*key* = *command_string*

Where *key* is replaced with the key combination that triggers the macro, and *command_string* is replaced with the command you want to execute when the macro is triggered.

For example, if you typed the following command, you would define the same function key macro you defined in the last lesson.

GLOBAL #KEY.AF1 = RULER

In fact, this is the syntax used by program to store the keyboard macro definition even when the KEY command is used to define the macro. Type the following command to see a list of all of the current keyboard macros in your cell:

SHOW USER=KEY.*

You can see that the macro assignment you made in the last lesson is stored using the syntax indicated above. Press both mouse buttons now to return to the view window.

When *key* is set to one of the special function key strings "F*n*", "AF*n*", "SF*n*", or "CF*n*", then the assignment is made to the combination of the indicated function key and the <Alt>, <Shift>, or <Ctrl> key. However, if one of these special *key* strings is not used, then the assignment is made to any ordinary keystroke combination.

For example, let us suppose that you need to add the text "VDD" to many places in your design. You could type the command "ADD TEXT='VDD' LAYER=TEXT SIZE=5" and then repeat it several times using the 1:Again menu option or the <↑> key. But suppose that you are executing other commands at the same time, and retrieving the ADD TEXT command each time from the command history is time consuming.

You could instead assign this command to the key combination <ATV> and the command would be executed every time you type those keys. Try this now by typing the following keyboard macro definition on the command line:

GLOBAL #KEY.ATV = “ADD TEXT=’VDD’ LAYER=M1 SIZE=5”

(Surrounding the command string with quote characters is recommended. Since this command uses quotes in the command, the use of two different quote characters is important. When your command string uses quotes, use a different type of quote character to surround the entire string, or your command will be misinterpreted.)

‘
“
`
~

Figure 33: Quote characters

Now type the keys <**A**> <**T**> then <**V**>. (The case is irrelevant.) You do not even need to press <Enter>. You can see the ADD TEXT command echoed on the command line. The editor is waiting for you to press the left mouse button and place the text component.

You do not need to use as many as three characters for the *key* string. You can assign a macro to a single character. However, when the key string is so short, it is easier to trigger it unintentionally. To reduce this problem, keyboard macros are triggered only when the key or key combination is the first thing typed on the command line. You cannot define a keyboard macro that is triggered while part way through typing a command on the command line.

Let us define a single character keyboard macro now. Let us suppose that you will be performing a MOVE X 10 operation on different sets of selected components at various places in your cell. You could repeatedly use separate operations to unselect everything, select a combination of components with SELECT IN, then issue the MOVE X 10 command. However, we will instead combine these operations into one command string and assign it to the single keystroke <M>. Type the following:

GLOBAL #KEY.M="UNSEL ALL; SEL IN; MOVE X 10"

In this case the quotes are required. If you did not surround the command string with quotes, then the command interpreter would see “GLOBAL #KEY.M=UNSEL ALL;” as the first command on the line, then “SEL IN” would be the second command. The SEL IN would not be interpreted as part of the macro definition.

However, since all three commands are surrounded by quotes, then all three commands are stored in the command string for the keyboard macro. All three commands will be executed when the keyboard macro is triggered.

To test the keyboard macro, you need some geometry in the cell. If the cell you are editing now has no components yet, add several boxes with the menu option:

1:(ADD)**box**

Now test the keyboard macro by typing <**M**>.

Why is nothing happening yet? Because the editor is waiting to see if you intend to keep typing to enter "MOVE…" or "MERGE…" or even "MENU…". When the keys typed on the command line are ambiguous (i.e. the key strokes could be interpreted as either the start of a command or as the trigger keys of a keyboard macro) then the editor will not trigger the macro until you type one more key that makes the key combination unambiguous. If you press <Enter> or press the space bar at this point, then the "M" is unambiguous and the keyboard macro is triggered. Press <Enter> now.

First, the UNSEL ALL command is executed. Then the UNSEL IN command waits for you to select components with the selection box. So click on two points so that at least one component is selected. Finally, the MOVE X 10 command is executed and the editor is waiting for next command.

The keyboard macros created with this method are saved in the cell file in exactly the same way as keyboard macros created with the KEY command. The macros are saved only in the cells saved at the end of the edit session. To make the keyboard macro definitions in all new cells, you must put the keyboard definitions in the startup command file.

Command Files

A command file is nothing more than an ASCII text file containing ICED™ commands. The *@file_name* command is used to execute the commands in the file.

The @ED command, used in the basic tutorial on page 37, executes the ED.CMD command file supplied with the installation. This command file uses the SHOW command, and other commands, to allow you modify component definitions.

A command file can contain many of the advanced features described in the Command File Programmers Reference Manual (including conditional execution macro definition, and even execution of other programs), or a command file may contain only a single command line.

We will now create a simple command file. Open a text editor to create a file with the name “MYCMD.CMD” in the working directory. If you are not familiar with Windows text editors, you can open an editor right from the ICED™ command line with the following command:

DOS NOTEPAD MYCMD.CMD

Now type the following line in the text editor window:

ADD WIRE LAY M1 W=3

Exit the editor by typing the following keystrokes:

<Alt><F>

<S>

<Alt><F>

<X>

Back in ICED™, type the following on the command line:

@MYCMD

You are now executing the ADD WIRE command in the file. Complete the command by digitizing vertices with the left mouse button, then pressing the right button to finish the wire.

You can assign this command file to the <F3> key with the following command:

KEY F3 = @MYCMD

Now any time you type the <F3> key, you will be adding a wire with a width of 3 on the M1 layer.

You could add other commands to the MYCMD.CMD file, and they would all be executed as though you just typed them at the command prompt.

ICED™ was able to locate the command file for execution since it was located in the working directory. However, cell libraries are not the best place to store command files. If you will be making use of command files, store them in a separate directory and add this directory to the command file search path described on page 66.

Many useful sample command files are supplied with the installation. Look in Q:\ICWIN[23]\TECH\SAMPLES to see other examples. You can also read the tutorial that explores this subject in detail beginning on page 223.

Log Files

ICED™ stores the commands excuted in the current session in a log file, *cell_name*.JOU in the working directory. This file is renamed to *cell_name*.LOG when you exit the editor normally. If you want an easy way to create a command file, you can execute the required operations with ordinary commands (or menu selections) in an edit session, then edit the log file to collect those commands and store them in a reusable command file. An example of this process is included in the Command Files tutorial on page 226.

Repeating and Canceling Commands

The following is a review of how to repeat, cancel, and undo commands in the editor:

Operation	**Menu Entry**	**Typed Command**
Repeat last command	1:**Again**	< ↑ > <Enter> (If ARROW mode is set to PAN, then you may need to hold the <Ctrl> key down.)
Repeat earlier command	(no menu method)	< ↑ > repeatedly, then <Enter>
Cancel current command (or menu selection)	Press both mouse buttons	
Undo last non-view operation	1:**UNDO**	UNDO
Undo last view window change	1:(VIEW)**last**	VIEW LAST

[23] Remember that Q:\ICWIN represents the drive and path where you have installed ICED™.

Undoing Larger Mistakes

The UNDO menu entry or command "undoes" only the last non-view window operation. What about when you need to undo the effects of the last several commands? In this case you need to use the journal (or log) file.

The basic idea is as follows:

1. Use the JOURNAL command to terminate the editor. This will **not** save the cell file(s) or rename the journal file from *cell_name*.JOU to *cell_name*.LOG.
2. Edit the *cell_name*.JOU file with a text editor to remove the first command executed in error **and all commands that follow until the end of the file**.
3. Open the layout editor as usual to edit *cell_name*. Since *cell_name*.JOU exists, ICED™ will provide you with the option of automatic recovery. When you reply with a Yes to the recovery dialog, the commands in the journal file will be executed automatically. This will recover all work done up to the command before the error.

There are variations on this process. You can even use a slightly modified process to undo only some of work performed in the last edit session when you have already saved the cell files. An entire tutorial covers this subject beginning on page 301.

Conclusion

You are now an expert on the methods used to enter commands in ICED™. Terminate the editor now with the menu entry:

1:**FILE → LEAVE**

To learn more about writing command files, read the Creating Useful Command Files tutorial beginning on page 223.

Selecting Components

In the basic tutorial, we covered the two main methods for selecting components for modification.

- Selection commands before the modification command
- Embedded selection (used when nothing is selected when a modification command is executed)

We will cover the embedded selection process only briefly here. Our main objective is to explore the selection commands SELECT and UNSELECT. These commands are used to build a selection of components before executing a modification command. We will explore the commonly used features of these commands. For the complete scope of the how these commands are used, you should read the SELECT command description in the Layout Editor Reference Manual.

For this tutorial, create a new cell in the TUTOR directory you have been using for the other lessons. You can clear the directory of its previous contents, but this is not critical. Open the console window with the ICED icon on your desktop, then change to the TUTOR directory and open the editor to create a new cell with ICWIN.BAT.

CD TUTOR

ICWIN SELCELL

Add a copy of the NAND cell with the following menu options:

1:(ADD)**cell → NAND**

(If the NAND cell is not on the first cell list, keep selecting NextPATH until it is shown.)

Now ungroup the cell (so that the cell is replaced with the components in the cell) with the menu options:

1:(SELECT)**new**

2:**EDIT**[24] **→ UnGRP**

Zoom in the view window with the menu option:

1:(VIEW)**all**

[24] Remember that to change to the second top-level menu you press the right mouse button or space bar.

Determining What is Currently Selected

ICED™ always reports how many components are currently selected in the command prompt. Note that the command prompt now contains the notation “Sel=0”. This indicates that no components are selected. Note that this parameter changes as you execute the following lessons.

Selected components are drawn in the same color as unselected components. The selection status is indicated by select marks (small white boxes) on the sides of the component.

Select all components and get a list of their definitions using the menu options:

1:**SELECT → ALL**

2:(SHOW)**screen**

Press the **<Enter>** key, or both mouse buttons, to return to the view of the layout. Note that the number of selected components is reported in the command prompt.

The selection status of each component is saved when a cell is saved. If components are selected when you save a cell, they will still be selected when you open the cell again at a later date.

Unselect all components now with:

1:(UNSEL)**all**

Embedded Shift Selection

Remember that when no components are selected before a modification command, ICED™ gives you the chance to select something appropriate before the command executes. If you hold down the shift key **before** you select a component with the mouse, you can continue to select multiple components for the modification command.

Insure that nothing is currently selected by noting the (Sel=0) report on the command line. Now begin a move operation with the menu entry:

1:**MOVE → X&Y**

The cursor changes to a small white box. This is because you are now executing an embedded SELECT NEAR command. The box cursor is called the near-box. Place the near-box over the edge of a component, press and hold down either <Shift> key, and click the left mouse button.

The component is now selected. If the box intersects more than one component, all components with edges in the near-box are selected.

Continue to hold down the <Shift> key, move the cursor to a different component, and click the left mouse button again. Now this component is selected as well.

Release the <Shift> key and select one more component. (You always release the <Shift> key **before** selecting the last component. If you forget to release it, you can click in any empty part of the drawing to indicate that you have finished the embedded selection process.)

The cursor changes to an 'X'. As you move the cursor, note that an 'X' is left behind at the last location of the cursor. You defined the first point for the displacement vector when you defined the last position of the near-box. This is a major convenience.

To move components, you need to digitize two points to define the displacement vector. If the first point is the old location of one the components, you need to define only the second point, where the chosen point on the component will be relocated.

After moving the cursor away from the original position, click the left mouse button to define the second point. The component(s) are moved to the new location and unselected. Move the components back to their original location with:

1:**UNDO**

You can see that where you position the near-box when selecting the final component is important. Select the final component at a point on an edge that is convenient to use as a reference point for the move.

Are you stuck if you selected the last component in an inconvenient spot? (Or if you had to click in an empty part of the drawing to complete the selection process?) No, you can redefine the first point of the displacement vector by clicking the **right** mouse button. Try this now by repeating the MOVE command with the menu option:

1:**MOVE → X&Y**

Now use shift selection to select a combination of components. Continue to hold the <Shift> key after selecting the last component, and complete the selection process by clicking in an empty area. Move the cursor slightly and note the first point of the displacement vector is marked with the 'X'. Now press the **right** mouse button. The first 'X' disappears. Move the cursor to a corner of a selected component and press the left mouse button. The first displacement point is now defined. Move the cursor away and you will see that this point remains marked with an 'X'. Press the left mouse button again to define the second point of the displacement vector and move the components.

Move the components back to their original location with:

1:**UNDO**

Selecting/Unselecting Components by Layer

When you need to select component(s) for an operation like a move, the embedded select feature of the modification commands is usually the most convenient method to use. However, what if you need to operate on a more complex group of components? Or when the single component that you need to select is in a dense area of the design and is difficult to select with the near-box? In this case, it is necessary to select the component(s) with SELECT and UNSELECT commands before the operation.

Suppose that you need to move most of the components in the NAND circuit, but you need to leave the POLY wires unmoved. This is easily done.

First, select all of the components in the NAND circuit. Use the menu option:

1:(SELECT)**in**

Use the 'X' cursor to draw a box that surrounds all of the circuit. Press the left mouse button at each corner of a rectangle as shown in Figure 34. This command will select all components that are entirely inside of, or cross an edge of, the selection rectangle.

Note that all of the components are now drawn with select marks. Note the report of the number of selected components in the command prompt.

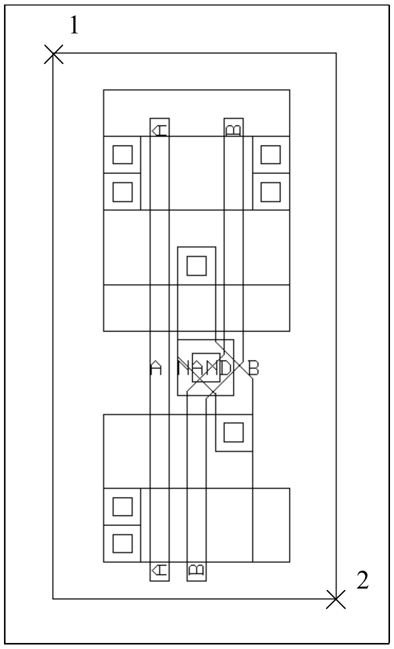


Figure 34: Using SELECT IN

Now unselect the POLY wires. Click on the menu option:

1:(UNSEL)**layer**

A submenu appears with a list of layer names. Place the cursor on the entry POLY and press the left mouse button. The layer is highlighted, but nothing happens. This is because you can highlight more than one layer. You can also use the options at the top of the menu to change the selection of layers. Now that the single layer we are interested in is highlighted, click on **Return** in the list, or press the right mouse button. Both methods complete layer selection and allow the command to continue.

Now a location sub-menu appears. Select **All**. You can see that the POLY wires are now unselected. You can also see the UNSELECT command generated from your menu choices on the history line near the bottom of the window.

Now move the components with the menu option:

1:**MOVE → Y**

You must digitize two points for the displacement vector. Use the cursor and left mouse button to do this. Note that displacement in the x-direction is ignored by the MOVE Y command.

We will reverse this last command and unselect all components with the menu options:

1:**UNDO**

1:(UNSEL)**all**

The layer restriction keywords are also available on the SELECT menu.

Types of Selection Commands

The layer test is one of 5 tests made to choose components for normal selection commands. The 5 tests are:

1.	Id test	(un)select components by unique id numbers
2.	Layer test	(un)select by layer
3.	Category test	(un)select by component type or edge type
4.	Status test	(un)select by current selection status
5.	Method test	(un)select by location or (un)select all eligible components

Several different tests can be combined into single (UN)SELECT commands. Components must pass all tests to be (un)selected. We will explore all of these types of tests in the following lessons.

The id test is rarely used. However it is the most restrictive. This option can be useful in especially dense layouts, however it is so rarely used that it is not available directly from the menu.

Only the more frequently used category test options are available from the menu. Display the entire SELECT sub-menu now by clicking on:

1:**SELECT**

Note the TEXT and CELL headings. The options under these headings restrict the command to these component types. Use CELL % options to select cells by name. Use the CELL * options to select all cells that pass the other tests.

The PARTS, SIDE, and END entries are also related to the category test. These options (un)select only certain sides of components. This is referred to as partial selection.

In addition to these "normal" selection operations, there is another class of selection commands that manipulate a set of selected components saved in a stack. The PUSH, POP, EXCHNG, and FAIL menu entries are used to manipulate the select stack.

We will be exploring all of these selection methods (and more) in the following lessons. Return to the first top-level menu by pressing both mouse buttons now.

Selecting by Component Type

You may have noticed in the lesson on page 97 that the text components that label the POLY wires were moved. It would be better to leave them unselected before the move. To do this, let us repeat part of the previous lesson with the following menu options. (Use the same selection rectangle that encloses the whole circuit.)

1:(SELECT)**in**

1:(UNSEL)**layer → POLY → Return → All**

Now we can unselect all text components with the option:

1:**UNSEL** → (TEXT)**all**

But suppose that we want the "A NAND B" text component to move with the rest of the components. We need to re-select it. Use the menu option:

1: **SELECT** → (TEXT)**near**

Now place the near-box over an edge of the "A NAND B" text component. Choose a place where one of the POLY wires is within the near box. Press the left mouse button. Note that the wire is **not** selected since the command was restricted to selecting text components.

Move the components with the following option this time:

1:**MOVE** → **X**

Now undo the move and unselect all components with the following options:

1:**UNDO**

1:(UNSEL)**all**

There are other component type restrictions you can add to (UN)SELECT commands. Refer to the ICED™ Layout Editor Reference Manual for a complete description and many more examples. Let us try one category test that is not available from the menus. You can restrict the components selected to wire components with the WIRE keyword. Type the following command:

SELECT LAYER=POLY+M1 WIRE IN

Draw the selection box around the NAND circuit and see that the boxes on M1 are not selected, only the wire components on layers POLY and M1.

Layer Lists

When you use the menu options to select layers in an (UN)SELECT command, you can build a list of layers for the select operation. You can also build a list of layers in a command typed on the command line. In fact, you did this with the last command indicated above. The parameter string "LAYER=POLY+M1" indicates that components on both the POLY and M1 layers will be considered.

The following syntax can be used to indicate layers or lists of layers any time you type a ICED™ command. An example command line is provided for each type of syntax.

The layer name (Abbreviations are valid as long as they contain enough characters so the abbreviation is not ambiguous.)

SEL LAY=PO ALL

The layer number (Note that the '=' after the LAY keyword is optional in all layer lists.)

SEL LAY 5 ALL

A colon (:) can be used to indicate a range of layers.

SEL LAY 100:255 IN

An asterisk (*) can be used in a layer range specification to indicate the highest or lowest valid layer number. (This next example selects the same layers as the previous example.)

SEL LAY 100:* IN

The '+' symbol is used to add layers to a list.

SEL LAY 1:5 +99

The '-' symbol is used to remove layers from a list.

SEL LAY 1:100 –M1

When used alone, the '-' symbol indicates the list of all layers *except* the indicated layer.

SEL LAY –M1

Selecting Parts of Components

Add all of the components in the bottom half of the circuit to the list of selected components with the option:

1:(SELECT)**in**

Draw the selection rectangle as shown in Figure 35.

Now we have the components on the bottom half of the circuit and all of the wires fully selected. Unselect the tops of the wires with the menu option:

1:(UNSEL)**side**

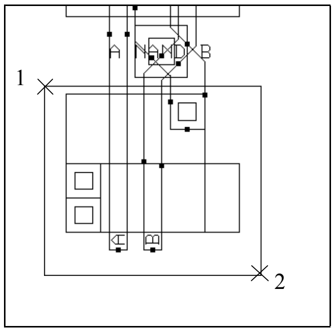


Figure 35: Using SELECT IN on bottom of circuit

Now draw the selection box around the top of the circuit as shown in Figure 36. After you digitize the second point, note how only the lower sides of the wires are now selected. We refer to components that have only certain sides selected as partially selected.

To fully select the entire components again, try the menu option:

1: **SELECT → PARTS**

This fully selects components that are partially selected. However, we want the wires partially selected as they were before, so now try:

1: **UNDO**

Note that UNDO does indeed undo the last selection command. UNDO can always be used to undo the result of the previous selection command.

Now move the lower half of the circuit down, while stretching the wires that connect the two halves, with the menu option:

1:**MOVE → Y**

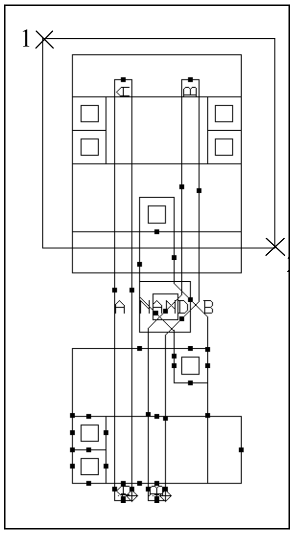


Figure 36: Using UNSELECT SIDE IN

Digitize the second point for the displacement vector below the first so that the coordinate report at the bottom of the window indicates a negative *disp_y* value.

(*current location*) -(*first coordinate*) = (*disp_x, disp_y*)

The final step in this process is to unselect all components with the menu entry:

1:(UNSEL)**all**

Depending on how you drew the selection boxes, the text component and via structure in the middle of the circuit may have been moved or not. Let us assume that you are not happy with the results of the move and want to undo it and try again. Try to undo the move with the following menu option:

1:**UNDO**

Note that the results of the move are not reversed, only the last UNSELECT ALL command. Try to UNDO again. The UNSELECT ALL command is re-executed. It is too late to undo the MOVE command. The moral of this is to always inspect the results of an operation **before unselecting** the components. Once you have unselected the components, you cannot undo the operation that preceded the UNSELECT command.

Making Components Unselectable

It would be easier to select only certain components in a layout if you could prevent others from ever being selected. This is easily done with ICED™. There are two commands that make components unselectable: PROTECT and BLANK. BLANK commands are very similar to PROTECT commands, but BLANK commands also make the components invisible.

Either command can operate on two different classes of components:

- Individual components
- All components on certain layers

For this lesson we will learn about protecting (but not blanking) individual components. You will select the individual components before executing the PROTECT command. Select only the M1 boxes that represent the metal power and ground busses with the following command typed on the command line:

SELECT LAY=M1 BOX IN

Draw the select rectangle around the entire circuit. Note that only the 2 M1 boxes are selected. Now protect these components with the following menu option:

3:**PROTEC → SELECT**

Now try to select one of these boxes with the near-box using the menu option:

1:**SELECT → NEAR**

No matter how you try to select either the M1 box, you will not be able to do so. Now you could edit the circuit for a long time and be absolutely sure that the shapes that represent the power and ground busses would not be changed. The components are unselectable by any method. This includes commands with embedded select operations. It also includes commands that do not modify geometry, like the SHOW command.

If you need to select the components, you must first unprotect them. Then make sure that no components remain selected.

3:**UNPROT → ALL**

1:(UNSEL)**all**

Making Subcells Unselectable

When you are adding or modifying dense wiring over or around dense rows of subcells, it would be very useful to temporarily protect all of the subcells so that you do not accidentally shift any of them as you change the wiring. To demonstrate this feature, you need to add a few more copies of the NAND cell to the drawing. Their exact location is not important.

1:(ADD)**cell → NAND**

1:**Again**

If you need to zoom out the view window to place the cells, remember that you can use the nested view menu with the <Esc> key during any command.

Now select all subcells in the drawing with the menu options:

1:**SELECT →** (CELL *)**all**

Remember that the options under the CELL * heading select cells with any name. Note that some structures in the ungrouped NAND cell were selected as well. See what these structures are with the option:

2:(SHOW)**screen**

There is one cell with the name VIA and several with the names NN and PP. These cells contain via or contact structures. Many layout designers use cells like these to connect different layers. For example, the VIA cell contains the minimum size via hole with the minimum overlap of the both metal layers so that the designer does not have to carefully lay out each via structure to be consistent with the technology groundrules. Also, if these groundrules change, the designer can edit the VIA cell itself and change every via structure in the design.

However, for our exercise, this use of cells in the wiring is awkward. How can you edit the wiring if all of the vias are unselectable? We need to unselect these types of cells before protecting the rest of the cells. This is easily accomplished.

1:**UNSEL →** (CELL %)**all→ VIA**

If you needed to, you could repeat the unselect procedure for the NN and PP cells as well. However we will not do this now. Protect the cells that remain selected with the option:

3:**PROTEC → SELECT**

Now the via cells in the drawing can be selected and moved or deleted as necessary, but all of the subcircuit cells cannot be altered. If you closed the editor and saved the cell file at this point, the subcells would remain protected the next time you opened the cell.

For now, let us unprotect all components before we move on to the next lesson.

3:**UNPROT → ALL**

Selecting New Components

The SELECT NEW command is used to select components that are “new”. What is “new”? Components just created by an ADD command are marked as “new” Components are also marked as “new” by any modification operation. Any modification command or ADD command will remove the “new” status of all components and then define only the components affected by the command as “new”. View commands and all other commands that do not involve selecting components do not change a component’s “new” status.

For example, try the SELECT NEW command now with the following menu option:

1:(SELECT)**new**

Note that the components that were unprotected in the previous lesson are selected.

In an earlier lesson on page 45 you used the SELECT NEW command to select the newly ungrouped components of a cell. Any time you need to select all of the components affected by the previous command, you can use 1:(SELECT)**new**.

Selecting Single Components in Dense Areas

Occasionally a component, or side of a component, can be difficult to select without selecting other components in the same area. We have already covered how to restrict an (UN)SELECT command by component type or layer. This is often enough to let you select the single component in question. However let us explore some more difficult examples to see how certain methods can make these types of selection easier.

Selecting a Single Side of Abutting Shapes

For this example, we need two boxes that abut each other along one side. We will then select a single side of the abutting pair for a move operation. Add the boxes by using the following menu entries so that the boxes look similar to Figure 37. The exact size or location is not important.

1:**ADD → BOX**

1:**Again**

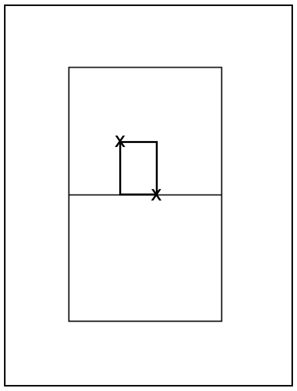


Figure 37: Select single side of abutting sides

If the fill mode is on, making it difficult to see the abutting edges, toggle the fill mode off with the menu entry:

3:(FILL)**tog**

To select only the bottom side of the top box with a SELECT IN rectangle, use the following menu entries:

1:(UNSEL)**all**

1:(SELECT)**side**

Digitize the selection rectangle so that the lower side is over the shared edge, and the rest of the rectangle is over **only the inside of the top box** as shown in Figure 37. This will select only the single side. Now move this side away from the other box with the following menu entry:

1:(MOVE)**side**

Digitize the two points of the displacement vector and only the single side should move. If both sides move, undo the move operation, unselect everything and try again after zooming in the view window with the menu entry:

1:(VIEW)**in %**

Selecting Components by Id Number

Each component is assigned a unique id number as it is added to the layout. You can use the id number to select a specific component. This method is most useful when two identical components are on top of each other.

Select one of the boxes now for a copy operation. If the single side is still selected, select the entire component with the menu entry:

1:**SELECT → PARTS**

Otherwise select a box with any convenient method. Now make another copy of the box on top of the first with the following command typed at the command line:

COPY 0,0

To determine the id numbers of the boxes, you must select both of them, then display their component information with the SHOW command. Using the SELECT IN operation, draw the selection rectangle so that it intersects only the original box and the copied box.

1:(SELECT)**in**

2:(SHOW)**screen**

The view window changes to display the component information on the selected components. The two components should be virtually identical except for their id numbers. Note one of the id numbers. Type <Enter> to return to the normal view window and then unselect everything with the menu option:

1:(UNSEL)**all**

Type the following command at the command line, where *nn* is the id number you noted above:

SEL ID=*nn*

Now delete the selected component with the menu entry:

1:**DELETE**

Now execute the same 1:(SELECT)**in** operation you did above that selected both boxes. This time only one box should be selected. The other identical box has been removed.

Using SHOW @ONE

Looking up the id number with the SHOW command can be cumbersome. There is a powerful new menu entry on the ICED™ menus that automates this process. The 2:(SHOW)**@ONE** menu entry executes a command file (SHOW1.CMD) that builds a list of all selected components and allows you to cycle through the list one component at a time. Each component will be selected individually and the SHOW information will be displayed at the bottom of the screen.

This menu operation can be used in two different ways. When one component is selected, the component information is reported on the history line on the bottom of the screen and the command is complete. When several components are selected, ICED™ builds a list of the selected components, unselects everything, and then selects the first component on the list and displays information about it. But the command is not complete. A sub-menu appears allowing you to cycle through the list of components one at a time.

If no components are selected when 2:(SHOW)**@ONE** is executed, an embedded SELECT NEAR will allow you to select something. If you position the near box to select a single component, the command will continue with the first method. If you select more than one component with the near-box, then the second method will be used.

Let us explore the first method of using this option. Unselect all components, then select a single component. Before you execute the 2:(SHOW)@ONE option, be sure that you have selected a single component by looking at the number of selected components reported in the command prompt.

> 1:(UNSEL)**all**
>
> 1:(SELECT)**in**
>
> 2:(SHOW)@**ONE**

Next you will execute the same series of commands again, but first we will add several copies of a cell on top of each other so that you select several components at once. Move to a clear section of the drawing and add several copies of the VIA cell. After adding the first copy of the cell, do not move the cursor. Just keep clicking the left mouse button to add the extra copies at the same exact location.

> 1:(ADD)**cell → VIA**
>
> 1:**Again**
>
> 1:**Again**

By looking, you cannot tell that there is more one via cell there. (This may seem like an unlikely scenario, but occasionally command files, or editing mistakes, can result in situations just like this.) When you execute the embedded SELECT NEAR command in the SHOW@ONE operation below, click on edge of the stack of via cells. This time the SHOW@ONE operation brings up a submenu since a list of components was selected.

> 1:(UNSEL)**all**
>
> 2:(SHOW)@**ONE**

Note that only one of the components in the list is now selected and it's component information is reported on the history line. Note that a new menu is displayed. (We will refer to options on this special menu with the prefix "S1:".) From this menu you can perform the following operations:

- delete the selected component with S1:**DELETE**,
- change the view with the options under S1:**VIEW**,
- go on to the next component in the list with S1:**NEXT**,
- return to the first component in the list with S1:**FIRST**,
- free the list and return to the regular menu (leaving any single component from the list selected) with S1:**RETURN**,
- free the list and return to the regular menu after unselecting all components with S1:**unsel**,

  or
- rotate to any the regular menus without freeing the list. Use the right mouse button to rotate the menus as usual.

If you use the last operation, one of the components in the list will remain selected allowing you to manipulate it with any of the other menu options. After executing any menu option, you will be returned to this special menu allowing you to move on to the next component in the list. The special menu that allows you to proceed though the components in the list is available until you free the list with the S1:**Return** or S1:**unsel** options, or exit the editor.

Delete each copy of the VIA cell and move on to the next component in the list with the options:

S1:**DELETE**

S1:**NEXT**

S1:**DELETE**

S1:**NEXT**

S1:**DELETE**

After deleting the last copy, you realize that you meant to keep one copy. You can rotate to the first regular menu and undo the last DELETE command.

1:**UNDO**

Now that you are back at the special SHOW@ONE menu, try using the S1:NEXT option:

S1:**NEXT**

The history line now tells you that you reached the end of the list. Return from the SHOW@ONE operation now by clicking:

S1:**RETURN**

Using the Select Stack

There may be times that you need to execute several SELECT and UNSELECT commands to select the set of components you need for a particular operation. Suppose you have your set of components selected just as you need them, then you remember that you need to perform some other operation first that requires you to select a different component. Do you need to unselect everything, perform this other operation, and then repeat all of the SELECT and UNSELECT commands to get your set of components re-selected? No, you can save a set of selections in a stack.

Let us demonstrate how this works. Explore with the SELECT and UNSELECT commands to select several components in the layout. Now execute the following menu option:

1:**UNSEL → PUSH**

This command will unselect everything in the layout after saving the selection status of the currently selected components. Note that 0 components are selected after the command is executed. You can perform any combination of commands now before restoring the original selection status of the components. You could even close the editor and restore the selections in a future edit session.

Select some other set of components with a SELECT menu option. Now unselect these components and restore the selection status of the original components with the following menu option:

1:**UNSEL → POP**

First all components are unselected, then the original components are re-selected. You could perform other operations, including UNSELECT commands, and reselect the original set again with the same pop command.

You can save only one set of selections in the stack. Every push operation replaces the contents of the stack. Any information saved by an earlier push operation is lost.

You do not need to unselect all components to push a set of component selections onto the stack or pop a set off. The PUSH and POP options are available on the SELECT sub-menu as well.

Select a different set of components with SELECT commands. Now push this set onto the stack with the menu option:

1:**SELECT → PUSH**

Note that the components remain selected with this option.

Unselect all components and select a single component that was not in this set of selected components with the following menu options:

1:(UNSEL)**all**

1:(SELECT)**in**

Now execute the following select stack operation:

1:**SELECT → POP**

In this case, the component selected before the stack operation remains selected. Now this component and the saved set of components are all selected.

Another select stack operation that is available on the SELECT sub-menu is the EXCHNG option that exchanges, or swaps, one set of selections for another. This operation is similar to a combination of a push and a pop. The effect of this command is to save in the selection stack the status of the currently selected components, then select only the selection set that was previously saved in the stack.

Selecting Failed Components

The FAIL option of the SELECT sub-menu will first push the current selections onto the stack. Then all components are unselected. Finally, the component (or components) that caused the previous command to fail will be selected.

To demonstrate the SELECT FAIL command, we must get a command to fail. When you try to move one side of a box past the far side of the same box, the move command will fail.

Select the bottom of the ungrouped NAND circuit with the following menu option. Be careful to digitize the second point carefully so that the right side of the M1 box is left outside the selection rectangle as shown in Figure 38.

1:(SELECT)**side**

Components entirely inside the selection rectangle are now fully selected, but the M1 box is only partially selected. Now move the selected part of the circuit to the right with the following command typed on the command line:

MOVE X 25

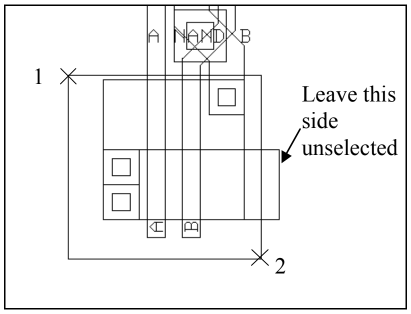


Figure 38: Using SELECT SIDE IN on bottom of circuit

The command will fail. It is not immediately obvious why it failed. The message on the history line gives you a clue, "****Move failed for 1 component(s)****". To see exactly which component failed, use the menu option:

1:**SELECT → FAIL**

The 3 sides of the M1 box are now the only items selected. You can now select the unselected side with:

1:(SELECT)**side**

Now restore the other previously selected components, without unselecting the side in question, with:

1:**SELECT → POP**

Now repeat the MOVE command by pressing the < ↑ > key until the MOVE X 25 command is on the command line. Then press <Enter>. The components are now moved successfully. The results look a little odd because some of the wires were only partially selected. Try the move again after you fully select all components with the following options:

1:**UNDO**

1:**SELECT → PARTS**

After you have successfully re-executed the MOVE X 25 command using the < ↑ > keys, reverse the effects with the option:

1:**UNDO**

One Final Caution

For our last lesson, move the view window to a clear section of the drawing. Now add a box on layer NDIF and select it with the options:

1:**UseLay → NDIF → BOX**

1:(SELECT)**new**

Now we will change the layer of this box with a SWAP operation.

2:(SWAP)**layers → NDIF → PDIF**

If your view window does not show the circuit we worked on in the last lesson, zoom out the view window until it does. Unless you caught on to our trick, you just changed the layer of the NDIF box in that circuit too. This happened because you did not unselect everything at the end of the last lesson, and those components remained selected during the swap operation.

You must always be very careful to unselect all components at the end of every operation and/or check that no components are selected by looking at the Sel=0 note in the command prompt before performing an operation on selected components. This is especially true when you change the view window away from the area where you have been working. Selected components do not have to be shown in the view window to changed by a command operating on selected components.

Conclusion

You now know how to use SELECT and UNSELECT commands to choose components for ICED™ operations. To see more examples, read about the SELECT and UNSELECT command in the ICED™ Layout Editor Reference Manual.

More on Creating and Modifying Components

This tutorial focuses on using the ADD command to create various types of components and on more advanced methods of manipulating components.

Since we will be exploring layer definitions, it is best to edit a cell created with the settings in the startup command file supplied with the installation. Open a console window with the ICED icon and change to the TUTOR directory. Launch the layout editor with the ICWIN.BAT batch file supplied with the installation to create a new cell with the following command:

```
ICWIN MORETUTR
```

The UseLay Menu Option and the Default Layer

The default layer is the layer to which components are added by an ADD command unless you override the layer specifically in a typed ADD command. When you use the options available on the menu to add a component, the component is always created on the default layer. There are three methods used to change the default layer:

- **USE LAYER command** The USE command is also used to set several other defaults used by the ADD command including the default text justification, the end type of wires, etc.
- **1:UseLay menu entry** This menu option combines a USE LAYER operation with an ADD operation to allow you to create a component on the selected layer. The selected layer remains the default layer after the ADD operation is completed.
- **LAYER command** When this command is typed with a single parameter of a layer name or number, it will report the current layer settings and make that layer the default layer.

The LAYER command has many parameters that allow you to change the settings of the layer (e.g. name, default width, display and plot colors and patterns). We explore this command in more detail in the next lesson.

You can read more about options available when typing USE commands in the Layout Editor Reference manual. The more common options (*except* for the USE LAYER option) are available in the submenu selected with the 1:USE menu entry.

The USE LAYER operation is selected with the separate 1:UseLay menu entry because it is so convenient to combine the default layer operation with the ADD component operation. Use this option now to set M1 as the default layer and create a box with the default width assigned to that layer.

1:**UseLay → M1 → Box**

Click the left mouse button to define opposite corners of the box. Note that the box is drawn with a cyan color and a pattern of slanting lines. These are the properties of the M1 layer set in the startup command file. Note that M1 is the default layer by looking at the command prompt.

Now change the default layer without creating any components with the following menu options:

1:**UseLay → M2 → MAIN**

Now add a polygon on this new default layer with a separate ADD command.

1:(ADD)**poly**

Digitize each vertex of the polygon with a click of the left mouse button. Create the polygon so that it overlaps the M1 box you created earlier. Close the polygon by placing the cursor exactly on top of the white X that marks the first vertex and clicking the left mouse button once more. (Remember that you can also close a polygon by clicking the right mouse button as you would to indicate that you are finished digitizing a wire, but this method may occasionally fail to close the polygon as you would expect.)

The LAYER Command

Note that you can easily see the outlines of both shapes added in the previous lesson even though they overlap. This is because of the different patterns used to draw each layer. These patterns were assigned with the following LAYER commands in the startup command file:

LAYER 6 NAME=M1 WIDTH=3.000 SPACE=0.000 CYAN PAT=2 …
LAYER 7 NAME=M2 WIDTH=4.000 SPACE=0.000 BLUE PAT=3 …

The patterns assigned to each pattern number are defined with the sample stipple pattern file, SAMPLE.STI, provided in the installation. We explore more about defining and assigning patterns in Manipulating the Display on page 173

The complete list of properties set with the LAYER command is:

- Name
- Default wire width
- Default wire spacing
- Color
- Display pattern
- Plot pattern
- CIF export layer name
- STREAM export layer number and datatype

(We explore the STREAM settings later on in Importing and Exporting Data on page 205).

All of these properties can be changed with layer commands executed in individual cells. Change the color of the M1 layer now with the following menu option:

3:**LAYER** → **M1** → (SET)**color** → (COLOR)**7:** → (PAT'RN)**NoChg**

Note that the M1 box is now drawn in a different color. The default layer has not changed. You can verify this by looking at the command prompt. When you change layer properties with this form of the LAYER command, the editor does not assume that you want to begin adding components on that layer immediately.

You should be careful when changing layer properties in individual cells. The layer properties are saved in the cell file, but they are not automatically changed in the startup command file that sets these layer properties for all new cells. Also, the layer properties of other existing cells are not modified.

However, let us explore defining layer properties for a while, then in a following lesson we will show you how to export these new layer definitions to your own startup command file.

Defining New Layers

Let us define a new layer with the name M1BUS. Execute the following menu options:

3:**LAYER → By # →16: →** (SET)**name**

Now type the name using the keyboard. You can type in lower case or in upper case. Case is irrelevant in layer names.

M1BUS

You have not really created a new layer. Instead you have assigned a layer name to layer 16 which is one of many layers with default properties that have no name assigned to them. All ICED™ cells have 256 layers.

Most properties of the M1BUS layer are still the default properties. Note that these properties are reported on the history line. Let change the color, pattern, and default width of the M1BUS layer with the following menu choices:

3:**LAYER → M1BUS →**(SET)**color →** (COLOR)**4: →** (PAT'RN)**2:**

3:**LAYER → M1BUS →**(SET)**width →** (WIDTH)**5.000**

We could have set all of these properties in the same command with the following menu options. (The "…" indicate that the menu options continue on the next line. This is just syntax to fit a long command on a single line in the manual.)

3:**LAYER → By # → 16: →** (SET)**SetALL →** (LAYER NAME)**KeyBrd …**

… (CIF NAME)**NoChg →** (STREAM LAYER)**NoChg →** (WIDTH)**5.000 …**

… (SPACE)**NoChg →** (COLOR)**4: →**(PAT'RN)**2: →** (PEN)**NoChg**

The editor would prompt you at this point to type in the layer name and the command would be complete.

You could alternately type the entire command on the command line:

LAYER 16 NAME=M1BUS WIDTH=5.0 COLOR=RED PAT=2

None of these methods will make the M1BUS layer the default layer. However typing the following command will change the default layer and report the current properties of the layer on the history line:

LAYER M1BUS

Now add a wire on this layer with the menu option:

1:(ADD)**wire**

After defining several points with the left mouse button. Indicate that you have finished digitizing points by clicking the right mouse button.

Listing and Exporting All Layer Definitions

Let us assume that you are creating many customized layer definitions. The best way to do this would be to create or modify a cell typical to your new design, then execute many LAYER commands to get layer colors and patterns set in a way that optimizes their visibility. You would also modify the other layer properties like widths and export assignments.

Let us say that after doing this for a while that you forget exactly how you set the properties of the M1 layer. We have already shown you how you can get a report of a single layer with the LAYER command:

LAYER M1

But how about if you want to see the entire list of named layers and all of their settings? This is where you use the TEMPLATE command. Execute this command now by using the following menu option:

2:(TEMPLA)**screen**

The view window changes to a listing of all environment settings in the cell, including layer properties. You can scroll this listing with the scroll bar on the right side of the window. Note that these settings are formatted as executable commands. If you could export this listing to a file, you could use it as a startup command file to get similar settings created in all of your new cells.

This is easily done. Return to the view window and the menu by clicking both mouse buttons at the same time. Now execute the following menu option:

2:(TEMPLA)**file**

At the prompt, type in the file name as follows:

MORETUTR

The file MORETUTR.CMD is now created in the working directory. Whenever you execute this file in a new or existing cell, the layer properties (and all other environment settings) will be changed in the open cell. When the cell is saved, these settings are saved with the cell file.

If you wanted only the layer definitions to be executed, you could edit the MORETUTR.CMD file to delete all lines except for the LAYER commands.

Clearing Layer Definitions

Two lines that were created in the MORETUTR.CMD file by the TEMPLATE command in the previous lesson are very important:

```
LAYER *  WIDTH=2.0  SPACE=0.0  YELLOW  PAT=0   NO_PEN
INITIALIZE LAYERS 0:255
```

The first line sets the properties of all unnamed layers. The second line applies these properties to all 256 layers, stripping the names and all other unique properties from all layers. Executing these two lines together gives all layers the same properties, clearing any other previous definitions. Leave these two lines in any command file that creates layer definitions.

Type and execute the INITIALIZE command above now. (The LAYER * command is already in effect from the startup command file.) Note that the components are now all drawn in the same yellow color without fill patterns.

Restore the layer definitions saved in the last lesson by executing the command file you created. Type the following command:

```
@MORETUTR.CMD
```

Note that the components are now drawn as they were before the INITIALIZE command.

Adding Wires

When you add a wire, you digitize points along the center of the wire. The edges of the wire are one half of the width away from the centerline of the wire. Let us make the M1 layer the default layer and add a specific wire to this layer with the following commands making use of shortened keywords:

```
LAY M1
ADD W  0,0  0,30  30,30
```

You may not see the wire in the view window, so now change the view window to show the new component with the following menu options:

1:(SELECT)**new**

1:**VIEW→ SELECT**

Now add another wire on the M2 layer using the same coordinates, but change the default wire type before creating the wire. (You want to use the command history to retrieve already typed commands to execute the last two commands. Press the < ↑ > key to go back into command history. Be sure to replace M1 with M2 in the LAYER command.)

USE WIRETYPE=0

LAY M2

ADD W 0,0 0,30 30,30

This wire looks quite different from the M1 wire even though they were created using the same coordinates. The default width of the M2 layer is 4.000 while the default width of the M1 layer is only 3.000. The ends are different because each wire has a different wire type. The startup command file specifies a USE command that sets the default wire type to 2 (extended ends), but the command we executed above changed the default to wire type to 0 (flat ends) before we executed the ADD command. The default wire type you should use depends on your technology. Normally you set the appropriate default in your startup command file, and you should never need to change it.

You may want to change the default wire width of layers as you edit cells. You may add several wires using the default width, then add several more of a different width. In this case, change the default width of a layer using the LAYER command, or the 1:LAYER menu option.

However, what about when you prefer to leave the default alone but need to add a wire using a width different from the default? In this case, simply override the width in the ADD command. Let us add a M2 wire with a width of only 2.000 using the following menu options:

1:**ADD →** (WIRE)**w=% →** (WIDTH=)**2.000**

The default width of the M2 layer is not affected by the command above.

What about when you need to change the width of a wire already in the layout? In this case you need to use the @ED command file as described back on page 37.

You do not need to resort to @ED to simply move one or more segments of a wire. Let us add another wire using specific coordinates with a typed ADD command.

Type the following command. (Do not type the "…". This is merely a way to fit a long command line into the width of this page.)

ADD W (-30.0, 15.0) (-30.0, 30.0) (-20.0, 30.0) (-20.0, 5.0) (-5.0, 5.0) …
… (-5.0, 15.0) (-15.0, 15.0)

The wire should look like the wire shown in Figure 38. Now let us move the top side down using the menu entry:

1:(MOVE)**side → Y**

Select the top side near the point indicated as point 1 in Figure 39. Now move the cursor until the y displacement indicated at the bottom of the window indicates –13.0. Then click the left mouse button.

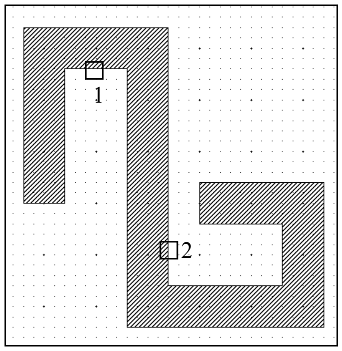


Figure 39: M2 wire

The wire is redefined without the redundant point at the left end. You can see this for yourself by reporting the coordinates of the wire on the screen with:

2:(SHOW)**screen**

Note that the length, perimeter, and area of the wire are also reported.

ICED™ always optimizes wire coordinates to remove redundant points. In addition, as you are digitizing points of a wire with the mouse, you can double back and shorten long segments, or add another point in a strait line from the last point and stretch a segment. You can even back up by redigitizing points in reverse. Try this now with additional ADD WIRE commands until you feel comfortable adding wire components.

You must be careful to avoid having wires cross themselves. Self-intersecting wires will cause problems for some post-processing software used in mask-making. To show that the layout editor will not prevent you from creating a self-intersecting wire, let us perform a similar move on the side indicated with the 2 in Figure 39.

1:(MOVE)**side → X**

Move the side at least 5 units to the right. You can see that the wire now clearly intersects itself. Avoid this type of self-intersection as you create and modify wires.

Using the Spacing Cursor

ICED™ has a visual aid that helps you place components together as tightly as possible, without violating minimum space requirements. You can change the appearance of the cursor to add guide marks at the minimum spacing distance. Line these guides up with existing components, and you know you are spacing wires (or other components) correctly.

To define the minimum space for a layer, you use the LAYER command. Let us use the following menu options to set the minimum space for the M1BUS layer.

3:**LAYER → M1BUS →** (SET)**space →** (SPACE)**2:**

Now turn the spacing cursor on with the following menu option

3:**SPACER →** (ON/OFF)**on**

You can set the minimum space between components with different several options of the SPACER command. We will turn on layer tracking which gets the minimum space from the SPACE property of the default layer. Turn this tracking on with the menu option:

3:**SPACER →** (SET)**track → on**

Now every time we change the default layer, the spacing will be set from the SPACE parameter of the new default layer automatically. Change the default layer to M1BUS and add a wire on this layer with the menu options:

1:**UseLay → M1BUS → WIRE**

Note the large circle around the cursor. Move this cursor near another component until it just overlaps it. The overlapping area is drawn in a different color, making it easy to see when the cursor is the correct minimum distance away. Follow the contours of one of the existing components to add the wire as closely as possible. Then complete the wire with the right mouse button.

There are 4 different spacing cursor styles. You are currently using style 1. Change the style and add another wire on M1BUS with the following options:

3:**SPACER →** (SET)**style → 3**

1:(ADD)**wire**

Try the other styles with other wires and see which one works best for you. Then turn off the spacing cursor with the menu option:

3:**SPACER →** (ON/OFF)**off**

Adding Polygons

Adding polygons is much like adding wires. You define vertices with the left mouse button. However with polygons, there are two methods to tell the editor you are finished digitizing coordinates and allow it to close the polygon.

- The best method is to redigitize the first vertex at a view scale fine enough to insure that the cursor is exactly on top of the first vertex.
- The second method is to click the right mouse button after you digitize the last vertex. (This is the same method as completing a wire.) If the editor can close the shape by extending existing sides, this will close the polygon.

First create a polygon on the default layer and close it by redigitizing the first vertex marked by the white X. Use the menu option:

1:(ADD)**poly**

Now add a second polygon. You may want to zoom the view window in with a VIEW command first so that you can digitize the points shown in Figure 40. Use the following menu options:

1:(VIEW)**in % → 2.0** (repeat as needed)

1:(ADD)**poly**

After digitizing point 3, swing the cursor around in a circle. Note how the cursor snaps to angles of 45º. Digitize points 4 and 5, then try to close the polygon with the right mouse button. Nothing happens.

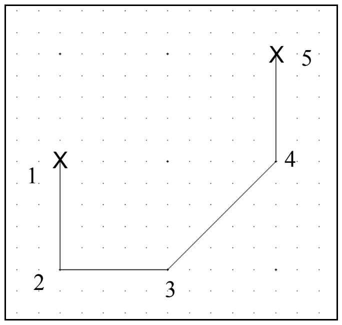


Figure 40: Digitizing a polygon

The layout editor cannot close the polygon by extending the first and last sides, so it cannot close the polygon. Digitize one more point above and to the left of point 5 with the left mouse button, then press the right button again. This time the right mouse button will close the polygon with a 45º angle.

Now create one more polygon with the menu option:

1:**Again**

This time, after digitizing point 5, redigitize point 4 and then point 1. Note that you can back up by redigitizing points already digitized.

Suppose that you need to know the area or perimeter of an oddly-shaped polygon. This is easily accomplished with the following menu option.

2:(SHOW)**screen**

Place the near box over an edge of one of the polygons and click the left mouse button. Note that in addition to the component definition, there are additional lines that display perimeter and area.

The RESOLUTION and SNAP Grids

In the previous lesson you were not prevented from adding sides at 45º angles from each other. You may prefer that all of your sides are at 90º angles.

An environment setting called the snap angle controls the angles allowed between sides of wires, lines, or polygons. Display the current value of the snap angle now with the following menu options:

3:**SNAP** → (ANGLE)**NoChg** → (STEP)**NoChg**

The history line near the bottom of the window should read:

```
SNAP ANGLE=45 STEP=(0.5, 0.5) OFFSET=(0.0,0.0)
```

This tells you that the snap angle is set to 45º, the snap step is set to (0.5, 0.5), and no snap offset is in effect. The snap offset and snap grid control which points can be used as valid coordinates. If you want to temporarily restrict points that can be digitized with the mouse to a course grid, perhaps a grid that is offset from the origin, you would modify these values. (The snap offset is so rarely used, it is not even included on the menu. You would need to type a SNAP command to set this value.)

Let us assume that you want to restrict angles between sides to 90º. Change the value of the snap angle with the following menu options:

3:**SNAP** → (ANGLE)**90** → (STEP)**NoChg**

You can also change the snap angle to 0. This allows sides to be at any angle. Creating and modifying components that contain arbitrary angles and skewed sides is covered in a separate tutorial beginning on page 145.

If you try to recreate the polygon in the previous lesson using the points in Figure 40, you will be unable to digitize point 4. The cursor will snap only to points at a 90º angle from the side ending at point 3. Cancel the command by pressing both mouse buttons.

The snap grid and snap angle control which points can be digitized with the mouse, and these environment settings are designed to be changed as required. The snap grid is based on a resolution grid that is designed to remain fixed throughout a project. It is set with the RESOLUTION command in the startup command file and should not be altered in individual cells. If you want to know more about the resolution grid refer to the Layout Editor Reference Manual.

Typing in Coordinate Data

Both of the grids mentioned above apply only to points digitized with the mouse. You can define components with coordinates not on this grid by typing the coordinates in an ADD command. Create a box with off-grid coordinates by typing:

ADD BOX (-30.001, 3.21) (-20.099, 10.965)

How many places to the right of the decimal can you define? This is controlled by a command line parameter variable called NDIV, which stands for number of divisions per unit. It is set in the project batch used to open the editor. NDIV defaults to 1000. So you can define coordinate data 3 places to the right of the decimal. If you need coordinate resolution finer than this, read about the NDIV parameter of the ICED™ command line in the Layout Editor Reference Manual.

Cutting Shapes

You can divide one shape along a straight line to create two shapes with the CUT command. Two options are available on the second top-level menu: 2:(CUT)**vert-X** and 2:(CUT)**hori-Y**. The first cuts a shape vertically along the line x=*coordinate*, the second cuts a shape horizontally along the line y=*coordinate*.

When you are cutting boxes or polygons, either option requires that the cut line intersects exactly 2 selected sides of each shape. Cutting a wire or line requires that the cut line intersects exactly one side of each shape. You can select the side(s) before the command with a SELECT command. If no components or sides are selected when the CUT command is executed, then the editor will execute an embedded SELECT SIDE IN command prior to performing the cut.

Let us try both methods. First let us cut the polygon created in the lesson on page 122 by selecting the sides first. Select the entire shape. If the command prompt already indicates that no components are selected with the comment "Sel=0", you can skip the first menu option.

1:(UNSEL)**all**
1:(SEL)**in**
2:(CUT)**vert-X**
1:(UNSEL)**all**

Position the cut line so it intersects 2 sides of the polygon and click the left button.

Now perform everything with the single menu option:

2:(CUT)**hori-Y**

Position the select rectangle as shown in Figure 41 so it intersects the vertical sides. Then position the cut line to intersect these sides.

Note that both shapes are cut. If other selected shapes (perhaps even shapes outside of the view window) were intersected by the same cut line, they would all be cut.

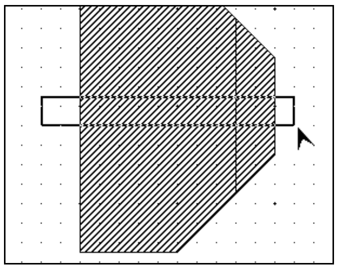


Figure 41: Selecting sides to cut

Adding or Removing Area from a Polygon

You can add or remove area from a polygon (or box) with the MERGE command. To combine polygons, the two polygons must share a common side, but no common area. To perform a merge of two portions of the cut up polygon from the previous lesson, execute the following menu option and position the near box over a common side of those portions.

2:(MERGE)**poly**

To remove area from a polygon, you need another polygon or a box whose area lies entirely inside the other polygon. This inner shape must share at least on common side with the outer shape. To see how this works, create a box in the lower left corner of the lower left polygon as shown in Figure 42.

1:(ADD)**box**

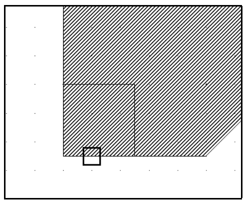


Figure 42: Box and polygon that share common area and sides.

Now remove the area of the box from the polygon with the following menu option. Be sure to position the near box over a common side as shown in Figure 42.

2:(MERGE)**poly**

The area of the box is etched away from the larger polygon.

Adding Text

ICED™ is very flexible when adding text components to a drawing. In fact, there are so many options on case, justification, appearance when rotated, etc. that we could devote an entire tutorial to adding text. However, we will cover only the most common options here. To learn more, refer to the TEXT and ADD TEXT commands in the Layout Editor Reference Manual.

We will add our sample text near the wires on M1BUS that you created earlier. To relocate the view window to display these wires, first display the entire drawing, and then zoom in on the area with the M1BUS wires with the options:

1:(VIEW)**all**

1:(VIEW)**box**

Now add text on the M1BUS layer with the options:

1:**UseLay → M1BUS →** (TEXT)**hori**

Type some text at the prompt, then press <Enter>.

Position the bounding box of the new component so it is aligned over of the wires, then click the left mouse button. Note that the text is slightly narrower than the width of the wire. This text was added using the default width of the M1BUS layer , as were the wires. Text components are automatically only 75% as tall as components of the same size so that the letters are not obscured by the outline of the component.

Now let us override this default size and add some larger letters on layer M1BUS.

1:**ADD** → (TEXT hori)**s=%** → (SIZE=)**10.0**

Type some text at the prompt and press <Enter>.

The TEXT command controls several settings that affect text components. Type this command with no parameters to see the current text settings.

TEXT

Note that the words "MULTI_LINE=DISABLED" are included in the report on the history line. Change that setting by typing:

TEXT MULTI=YES

Note that multiline text is now enabled. Now repeat the ADD text command we used earlier with the menu options:

1:**ADD** → (TEXT hori)**s=%** → (SIZE=)**10.0**

Type some text and press <Enter> as before. Now you get another prompt instead of the end of the command. Type some more text and press <Enter> again. You could add several lines in this manner. Press <Enter> on an empty line to finish adding lines. Now place the text component with the mouse.

Normally, you want to leave multi-line text disabled so that you do not have to press <Enter> twice every time you create a text component. Disable this feature again by typing:

TEXT MULTI=NO

You can also enable lower-case text with the TEXT command (or with the 2:TEXT menu option) but we strongly recommend that you do not. Lower case text can be a problem for some types of post-processing software.

One other important setting of the TEXT command is the default justification code. The text components we have added so far have used the setting of LB or lower-bottom for justification. This means that the origin of the text component is placed in the lower left corner of the bounding box.

The origin of the text component is stored as its location. If you will be using the NLE to extract a netlist from the layout, text justified in this manner is not recommended to label nets or devices. It is much better to use the center of the text component as its location so that the location is always completely covered by the component it is meant to label. To change the default justification for new text components, type the following command:

TEXT JUST=CC

Now add one more text component to see that the cursor is now located in the center of the new component's bounding box.

Modifying Text with @ED

The only easy way to change the text in an existing text component is to use the @ED.CMD command file we first described in the Layout Editor Basics Tutorial on page 37. Select a text component, then type:

@ED

A text editor window opens up displaying the ADD TEXT command for the component. Edit the text in quotes in some noticeable way then exit the text editor with <Alt><F> and then <X>. Type <Enter> at the "Save it now?" prompt. The original text component is deleted and the modified component is added at the same location.

Now change the component back to its original text by typing the following command:

@UNED

Adding Maskable Text

ICED™ text components are not maskable geometry and they are filtered out before using the layout data for mask generation. However, you can create real polygons or wires in the shape of alphanumeric characters that will be created on a mask. This is often done to create comments that appear on the physical semiconductor product.

You can create characters freehand, but there is an auxiliary program included with the ICED™ installation that will convert ordinary text components into polygons representing characters suitable for mask generation.

The name of the program is **PGTEXT.EXE** for **P**attern **G**eneratable **Text** creation. The program is more completely described in a file PGTEXT.TXT in your Q:\ICWIN\DOC directory. The source code (in Pascal) is included in its own directory, Q:\ICWIN\PGTEXT.

To use PGTEXT.EXE, you first create a text component in the layout editor. You then export the definition of the component using the SHOW command. The PGTEXT.EXE program converts the text into polygons. You import the polygons into your layout by executing a command file created by the program.

The following characteristics of the original text component are preserved in the created polygons:

layer,
scale (height is preserved, width usually extends somewhat),
justification,
rotation, and
location

Lower case text is not supported. It is translated to upper case before the polygons are created. Multiline text is also not supported. You can translate only a single line of text, stored in a single text component, with a single call to PGTEXT.EXE.

This program is most easily used when you surround the call to PGTEXT.EXE with a customized command file that will insure that only a single text component is selected. Commands that export the text component with the SHOW command, and then import the result created in the PGTEXT.CMD command file should be included. Read PGTEXT.TXT if you will use this feature extensively.

However, some simple automation for this feature is included with the installation. We will use this method for this lesson. We will assign a command string to a function key that automates most of the process. Execute the Q:\ICWIN\AUXIL\PG_KEYF7.CMD command file that assigns the appropriate command string to the <F7> function key by typing the following:

@PG_KEYF7

You should see the following command reported on the history line after executing the above command:

```
GLOBAL #KEY.F7="SELECT NEW; SHOW PROG=0 FILE=PGTEXT; …
… DOS '^PGTEXT'; @PGTEXT"
```

Now add a text component to the drawing with the menu options:

1:**ADD** → (TEXT)(hori)**s=%** → **10.000**

Type any string you like at the prompt and press <**Enter**> to add the text component. Then press the <**F7**> key.

The text component is exported, the PGTEXT.EXE program is called, and finally the PGTEXT.CMD file (created in your working directory) is executed to create the polygons.

Note that the polygons are somewhat taller that the text component. Remember that an ICED™ text component is drawn somewhat smaller than its indicated size so that the text is not obscured by the outline of the component it is meant to label. However, you can see that the height of the polygons is the 10 user units you selected when you created the text component.

Note also that the maskable text is somewhat longer than the original text component. This is due to the font used by the PGTEXT routines.

> You should make the original text component lines somewhat shorter than the total width of the space you have available for the maskable text.

UNDO will not reverse the effects of the <F7> key. You need to delete the polygons by hand if they were not created correctly. (If you create your own command file to use PGTEXT.EXE, you may want to include the ability to reverse its effects.)

You should add the @PG_KEYF7 command to your startup command file if you want to use this method in all of your cells.

Selection of Components Before and After PGTEXT.EXE

The original text component is not deleted by the command string executed by <F7>. However it is left selected so that you can easily delete it after looking at the results. Delete the original text component now with the menu option:

1:**DELETE**

When using the <F7> method above, you should be careful that no other components are selected before you press the key. The operation may do nothing if components other than a single text component are already selected. Only one text component should be selected. Alternately, you can unselect everything, and then create the text component immediately before pressing the <F7> key and it will be selected by the "SELECT NEW" command in the command string assigned to the <F7> key.

Special Copyright Characters

PGTEXT.EXE supports two special characters. To create one of the special characters shown in Figure 43, convert one of the two-character text component strings shown. The two characters must be the only characters in the text component. If other characters are present, then the string will not be interpreted as containing a special character.

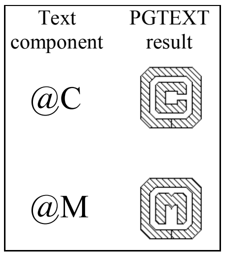


Figure 43: PGTEXT special characters

Insuring Characters Are On Grid

If you need all of the vertices in the polygons created by PGTEXT to be on grid, you must be careful when selecting the text size.

The font style in PGTEXT is based on a 10 micron height and 0.5 micron resolution. To insure that the generated text has all vertices on a grid with a given resolution, the following equation can be used to calculate valid text sizes:

Text_Size = *n* ∗ 20 ∗ *resolution*; where *n*=1, 2, 3, ...

Therefore, if your grid resolution is .5 user units, valid text sizes are 10, 20, 30, etc. If your grid resolution is .1 user units, valid text sizes are 2, 4, 6, etc.

Avoiding Design Rule Violations

You must take care that your characters do not violate minimum space or minimum width design rules. While these design rules may not really apply to maskable text (since no electrical circuits are formed), the DRC program (or any other design rules checker) will not automatically distinguish them from any other shapes on a mask layer. Ignoring lists of design rule violations for maskable characters is a dangerous practice since a design rule violation in real circuitry may be overlooked.

If you cannot avoid violating design rules with your maskable characters, devise a method to isolate these shapes from real circuitry on the same layer. The most common method is to use an AND NOT rule and a dummy shape to prevent shapes covered by the dummy shape from being copied to the layer used in the design rules. Just be careful that the dummy shape does not prevent minimum space violations between the characters and real circuitry from being found.

Protecting Layers from Changes

You can prevent components from accidental modification with the PROTECT command. We already learned how to protect selected components back on page 102. Now we will protect an entire layer.

3:**PROTEC → layers → M1BUS → Return**

Now try any way you can think of to select one of the existing M1BUS components for a modification command. You will not succeed since all components on the M1BUS layer are protected from changes. If M1BUS represented the layer used for power and ground busses, you would not be able to accidentally shift one of the bus wires and create many open circuits accidentally.

Now add another component on layer M1BUS. Try to select it. This new component you can select. To protect components on a protected layer that were added after the PROTECT LAYERS command was executed, type the PROTECT command with no parameters.

PROTECT

The components will remain protected even after you save the cell file and reopen it. You must unprotect the components to modify them. Do this now with the menu option:

3:**UNPROT → all**

Mirroring and Rotating Components

Select all of the M1BUS components in the layout with the menu option:

1:(SELECT)**new**

This selects all of the components unprotected by the last command in the previous lesson. Now we will rotate all of these components 180º counter-clockwise with the menu option:

1:**MOVE →** (ROTATE CCW)**r2=180**

A cursor appears in the view window waiting for you to define an axis point for the rotation. Move this cursor if you like, then click the left mouse button. Notice that all components are rotated. Note that all text components still read right-side-up and right-to-left. This is due to a setting of the TEXT command. Type the TEXT command with no parameters:

TEXT

Note that one of the parameters reads “ORIENTATIONS=2”. This setting will cause all text components to read either right-to-left or bottom-to-top. Change this setting now by typing the command:

TEXT ORIENT=8

Now the text appears rotated.

Unselect all components and add a new text component with the menu options:

1:(UNSEL)**all**

1:(ADD**)text**

Now copy this component and mirror it at the same around the line x=*coordinate* time with the options:

1:(SELECT)**new**

1:**COPY → MIR X**

Note that the copied text component now displays mirrored text. If you would prefer to see mirrored text components in an unmirrored state, change the setting of the text command with:

```
TEXT ORIENT=4
```

Click 1:UNDO a few times to see the difference.

Line Components

Line components are used for documentation rather than to create mask geometry. They can be useful as visual aids. They can be added, cut, moved, or otherwise modified just like wire components with 0 width.

Let us name a new layer for documentation notes and add a line to the drawing on this layer. As we saw in the lesson on the LAYER command on page 114, defining a layer can be easier with a typed command than with the menu entry if you are defining only one or two properties. So let us type the command to define this new layer.

```
LAYER 50 NAME=DOCS WHITE
```

Note that the COLOR= keyword can be omitted. The syntax parser recognizes white as a color name even without the COLOR keyword.

Use the 1:VIEW options or the arrow keys to change to an empty section of the drawing. Then create the line with the menu options:

1:**UseLay → DOCS → LINE**

Add a line that contains a horizontal segment at least 60 units long. Go off the edge of the screen an note that the window automatically pans. When you are through digitizing corners with the left mouse button, click the right mouse button to complete the command.

Adding Cells

Cells can be added with the menu option 1:(ADD)cell or by typing the ADD command. Typing the ADD command allows the cell to be mirrored or rotated before it is placed. We will try both options below.

To use the menu option, click on the following:

1:(ADD)**cell**

You are presented with menu of cells to choose from. The list displayed in the menu depends on what cells are available. If you have cells already added to the current cell, or edited in the current session, they will be listed in the first menu. The history line will report "*n* cells in use".

If you have no cells "in use", then the first menu will list the cells in the working directory. If there are no cells in the working directory, then the first menu will list the cells in the first cell library on the ICED_PATH environment variable described in the Customizing ICED™ for Specific Projects tutorial on page 53. In either case the history line reports the name of the directory where the cells were found. The line below reports how many cells are in the directory.

You always have three options at the top of the cell list:

• NextPATH	This option displays the cells in the next cell library.
• NextPage	If the current cell list has so many cells that they do not fit in the menu area, then you can see the next page of cells with this option.
• KeyBoard	This option allows you to type the cell name. All cell libraries are searched in order to find a cell with that name. If no cell with this name is found, you will get an error message.

If NAND is not listed in the first list, keep selecting NextPATH until it is, then click on NAND. The bounding box of the cell is displayed. Note that the cursor is **not** located in the lower left hand corner of the cell. The cursor is located on the origin of the cell. When you click the left mouse button, the origin of the cell is placed at the current location of the cursor, and this location is stored as the location of the cell.

Place the cell near, but not on top of the long horizontal line segment you created in the previous lesson. The click the left mouse button. Click on the following menu option to add another copy of the same cell.

1:**Again**

Place this copy at some distance vertically from the line. Move the cursor off the top of the window and the window will pan up. Place the cell with the left mouse button.

Now let us try the typed command:

ADD CELL NAND MX

Place the cell near the others, but do not overlap the cells. Note that this copy is mirrored about the line x=*coordinate*, where *coordinate* is the x-coordinate you digitized with the mouse.

We'll rotate the next copy of the NAND cell 90º counterclockwise about the origin of the cell with the command:

ADD CELL NAND R=1

Aligning Components

While you can move several selected components all at once with the layout editor, they all move by exactly the same distance. To align several components you must move each component separately. Let us align the unrotated NAND cells to the line component.

Use the VIEW options to zoom in around the bottom of the first NAND cell you added near the line component. The line component should also be in the view window. Make sure that no components are selected. Now to align this cell vertically, use the menu options:

1:**MOVE → Y**

You are now executing an embedded SELECT NEAR operation. To select a cell with this option, you must place the near box on the bounding box of the cell indicated with the dotted line. But don't click the left mouse button yet!

Where you select the cell is important. The location on the cell where you click the left mouse button is the first point on the displacement vector. Once you move the cursor and click the left mouse button again, the spot you selected on the cell will be moved to that second location. Except in this case, since you are using the MOVE Y operation, displacement in the x-direction is ignored.

Position the near box carefully so that it is centered on the lower edge of the box on M1 that represents the ground bus, **and** it is over the bounding box of the cell. Look at Figure 44. Now click the left mouse button. Now center the cursor over the line component and click the left mouse button again. The cell is moved vertically to align the bottom of the M1 bus shape on the line.

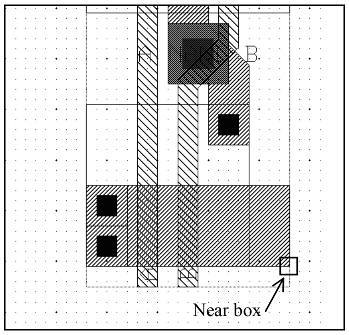


Figure 44: Selecting a cell

Now let us perform the same operation on the NAND cell that was farther away. Change the view window so that the NAND cell that is farther away and the line are both displayed. Now begin the MOVE Y operation again.

1:**Again**

At this scale you probably cannot choose the correct location on the cell so easily. We will use the nested view menu to zoom in on the bottom of the NAND cell. Click the center button of your mouse. (If your mouse has only two buttons, use the <Esc> key instead.) Now select the following nested view menu option:

BOX

Use the left mouse button to draw the box around the bottom of the NAND cell to zoom in. Now place the near box carefully and press the left mouse button. You are still executing the MOVE operation. You must change the view window to zoom in on the line. Use the nested view menu again, and this time use the nested view option:

LAST

If you cannot place the cursor carefully on the line at this scale, use the nested view menu again to zoom in. Click the left mouse button again once the cursor is on the line. The component is now aligned.

Suppose that we needed to align the NAND cells so that the via shape in the center of the cells was aligned with the line. Let us move another NAND cell to try this. Change the view window again if necessary to display another NAND cell and begin the MOVE operation with:

1:**Again**

Now try to select the cell on the via shape in the center of the cell. You cannot select the cell this way because the bounding box does not overlap this shape. However, we can select the cell, then redefine the start of the displacement vector on the via shape.

First, select the cell by placing the near box over any or the sides of the cell. Note the white X indicating the first point on the displacement vector. Now click the right mouse button. The white X disappears. Place the cursor over a spot on the via shape and click the left mouse button. The white X is left at this spot. Move the cursor to the line and press the left mouse button again. The via shape is now aligned with the line.

Adding Arrays

You do not need to add cells one by one and align them to get an array of cells. We will now add an entire array of cells with one command. First, zoom the view window out with the menu option:

1:(VIEW)**all**

Now add an array of NAND cells with the options:

1:**ADD → ARRAY → NAND →** (ROTATE CODE)**none →** (STEP)**Def'lt**

At the "Enter NX = number of columns" prompt type in:

10

At the "Enter NY = number of rows" prompt type in:

1

The bounding box for the entire array appears on the screen with the cursor positioned on the origin of the leftmost cell. Move the bounding box to an empty place in the drawing and click the left mouse button.

Note that the cells are abutted up against each other. This is because you selected Def'lt (or Default) for the array step. You could have selected (STEP)KeyBrd (or Keyboard) to define the distance from one cell to the next if you did not want to use the default of abutting the cell bounding boxes in the array.

The entire array is a component. You can only select the entire array at this point for modification operations (e.g. MOVE). We will be replacing some of these cells in the next lesson, so we will ungroup the array into cell components that can be modified individually with the menu options

1:(SELECT)**new**

2:**EDIT → UnGRP**

While the array looks the same, you can now select the cells individually and move or otherwise modify them.

If you add large arrays to your design, you may want to change how they are displayed to simplify the display of your design. You can display only the bounding box of the array, or only the cells on the outside border, or all cells. We explore the use of the 3:ARRAY menu option that controls how arrays are displayed in another tutorial on page 184.

Swapping Cells

In this lesson we will replace some of the cells in the array with the modified NAND cell you created back in the basic tutorial on page 46, NANDTEST. If this cell is not in the working directory any longer, just use a different cell for the swap.

First make sure no components are selected, and then select only some of the NAND cells in the array with the menu option:

1:(SELECT)**in**

To perform the swap, select the following menu options:

2:(SWAP)**cells → NAND → NextPATH → NANDTEST**

Note that the selected NAND cells are replaced with NANDTEST cells. Now select all of the cells in the array with:

1:(SELECT)**in**

Perform the same swap command by using the command history feature. Press the < ↑ > key until the swap command is on the command line, and then press <Enter>.

Not only were the remaining NAND cells replaced with NANDTEST cells, but the NANDTEST cells were swapped back to NAND cells. This is the way the swap command works. All selected occurrences of *cell1* are replaced with *cell2*, and all selected occurrences of *cell2* are replaced with *cell1*. Always be careful of what is selected any time you are using a swap command.

Changing a Component Layer

The SWAP LAYERS command works in the same way as the SWAP CELLS command. All selected components on *layer1* are changed to *layer2* Let us try this on the components in a NAND cell.

First let us select only one of the NAND cells, change the view window to zoom in on this cell and ungroup it into individual components. Use the menu options:

1:(UNSEL)**all**

1:(SEL)**in**

1:**VIEW → SELECT**

2:**EDIT → UnGRP**

Now we will swap the PDIF and NDIF layers in the ungrouped cell. First select all of the components of the ungrouped cell. It will not affect the SWAP command if components on unrelated layers are selected. Only the layers mentioned in the command will be affected. Use the menu options:

1:(SEL)**new**

2:(SWAP)**layers → PDIF → NDIF**

The order in which you select the PDIF and NDIF layers is unimportant. You could select NDIF first, then PDIF, and the command would execute in exactly the same manner.

You can see that the green and yellow boxes have changed colors. Let us verify that the yellow box in the lower half of the circuit is really on layer PDIF.

1:(UNSEL)**all**

2:(SHOW)**@one**

Now click on the yellow box in the lower half of the circuit. The history line will report the definition of this component. Note that the PDIF box remains selected.

Another method to change the layer of a component is to use the ED.CMD command file. We will use this command now to change the PDIF box back to layer NDIF. Type the command to execute this command file

@ED

In the text editor, change the "PDIF" to "NDIF". Type <Alt><F>, then <X>, and finally <Enter> to save the changes. Note that the box is now back on the NDIF layer.

Moving Sides

We have spent a lot of time changing component layers and locations. Now let us change the shape of some components. The 1:MOVE options are used primarily to move entire components, but if you select only certain sides prior to executing this option, only those sides will be moved.

As we learned in the tutorial on selecting components, you can combine selection criteria into a single typed SELECT command. Let us now select only the leftmost sides of the NDIF boxes of our ungrouped circuit with the typed command:

SEL LAY=NDIF SIDE IN

Draw the selection box as shown in Figure 45. Note that since the selection box crossed the horizontal sides of these shapes, they were also selected. However if we specify a horizontal displacement, that will not matter. Move the selected sides in only the x-direction with the menu option:

1:**MOVE → X**

You could have used the 1:(MOVE)side → X options to perform this operation and the result would have been exactly the same. The two options are different only in the embedded selection operation that is used when nothing is selected prior to the move operation.

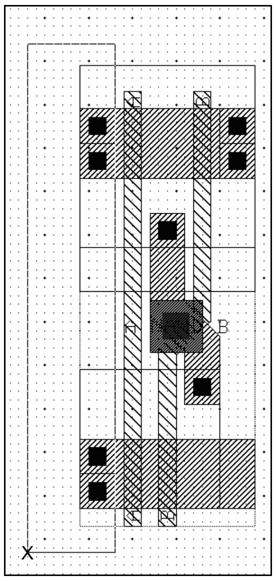


Figure 45: Selecting only sides on NDIF layer

- When you want to move several sides, select these sides first then use either 1:MOVE or 1:(MOVE)side.
- When you want to move an entire single component and nothing is currently selected, use 1:MOVE.
- When you want to move a single side and nothing is currently selected, use 1:(MOVE) side.

Let us try the third option now. First unselect all components, and then move a single side with the options:

1:(UNSEL)**all**

1:(MOVE)**side**

Select a side of a box with the near box then move the side by the desired displacement.

This is the same method you would use to move a segment of a wire. Use the 1:VIEW options to zoom in on one of the M1BUS wires. Now move one segment of one of these wires with the option:

1:(MOVE)**side**

Note that the other wire segments expand or contract as necessary to accommodate the new position of the selected segment.

Conclusion

If you want to see more examples of defining layers, look at the LAYER commands in the startup command file supplied with the installation, (Q:\ICWIN[25]\TECH-\SAMPLES\NEW.CMD) shown on page 49. If you need to learn how to assign stream numbers in these layer definitions, see the tutorial on exporting data on page 205.

If you want to learn even more about creating components, read the ADD command description in the Layout Editor Reference Manual.

[25]Throughout this manual, Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. If you have installed the software on your C drive in the directory \ICED, you should replace Q: with C: and \ICWIN with \ICED.

To learn about using Boolean operations on components, see the tutorial beginning on page 269. Many advanced component and layer manipulations are available on the special Internal DRC menu described in this tutorial. This includes an alternate method for merging components that does not require touching sides.

If you want to learn more about creating and modifying components with non-perpendicular sides, read the material in the next tutorial. This tutorial includes information on how to create components like rings, circles, and arcs.

If you will not be continuing with the next tutorial immediately, you should close the editor now with the menu option:

1:**FILE → LEAVE**

Creating Components at Arbitrary Angles

Just about all of the geometry you created in the previous tutorials has all sides at 90º to each other. In this tutorial we will look at creating and modifying components with sides at arbitrary angles to each other. You will create circular components and other components with skewed sides.

You should be aware that components with acute angles could be a problem for some types of post-processing software. ICED™ will not prevent you from creating shapes with acute angles. You should be familiar with the restrictions in effect for your process. The DRC (Design Rules Checker, also available from IC Editors) has several rules to find acute angles on mask layers.

Open the editor to edit the same cell used in the previous tutorial, or you can create a new cell. Use the Q:\ICWIN[26]\TUTOR subdirectory as your working directory.

Entering the Any-Angle Mode

To enter the any-angle mode, use the menu option:

3:**SNAP** → (ANGLE)**0°** → (STEP)**NoChg**

This sets the snap angle to 0° without changing the snap grid step. A snap angle of 0° places ICED™ in any-angle mode. Any angle between sides of a polygon, wire, or line is acceptable.

Now add a polygon using the menu option:

1:(ADD)**poly**

Notice as you digitize points that the polygon's sides are no longer constrained to 45° or 90° angles. After digitizing several points, close the polygon by redigitizing the first vertex.

[26] Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software.

Selecting and Moving Vertices

In this lesson, you will select and move a single vertex of a box. The slopes of the sides connected to the vertex will change and only the selected vertex will be moved. Make sure that no components are currently selected, and then create a box by using the menu option:

1:(ADD)**box**

Since the vast majority of geometry is created with perpendicular sides, moving sides is all that is supported on the menu when the snap angle is set to 45º or 90º. However, now that the snap angle is set to 0º, a new option has been added to the menu under the MOVE heading. To move a vertex, use the following: menu option:

1:(MOVE)**vertex**

You are now executing an embedded SELECT SIDE NEAR command. A vertex is considered selected if the two sides that intersect at the vertex are selected. To move one corner of the box, place the near-box cursor carefully on a corner of the box and press the left mouse button. White select marks should appear on two sides of the box.

Now move the cursor to where you want the vertex to end up on completion of the command by pressing the left mouse button. Your view window should look something like Figure 47. Notice that the resulting component is no longer a rectangle.

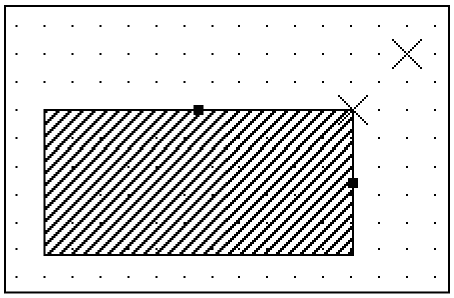


Figure 46: Using cursor to define displacement in MOVE VERTEX.

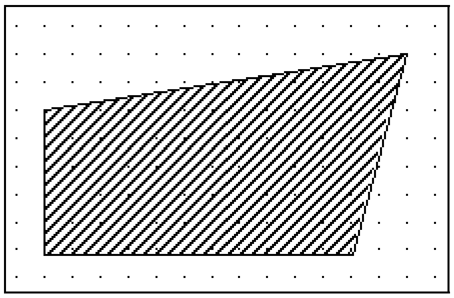


Figure 47: Result of MOVE VERTEX.

You can move a vertex of a wire or line using the same operation. Moving a vertex in this manner is possible even when the snap angle is set to 90º when you type the MOVE VERTEX command.

The MOVE SIDE Command and Any-Angle Polygons and Wires

When you move sides of a component with MOVE SIDE, the sides can only be shifted parallel to the original sides. The ends of the selected and unselected sides are extended until they meet. All vertices connected to the selected sides may move and there is no guarantee that the moved vertices will be on grid. Indeed, if you are stretching a polygon with slanted sides, the new vertices will only rarely be on grid. When the moved vertices are not on grid, the command will fail.

Create a polygon similar to the one shown in Figure 48. Then begin a MOVE SIDE operation. Use the options:

1:(ADD)**poly**

1:(MOVE)**side → X&Y**

Position the near box as shown in Figure 48 to select a vertex. Then use the cursor again to define the displacement to the new vertex location as shown.

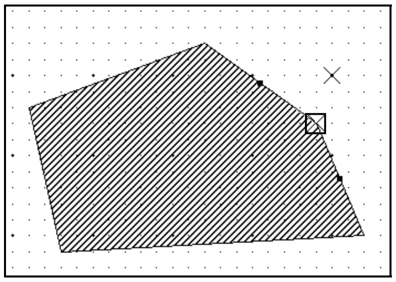


Figure 48: Using the cursor to indicate displacement of a vertex for MOVE SIDE.

If your polygon has slanted sides, the command probably failed with a warning message similar to "***** Move failed for 1 component ****". To see why the command failed, look at Figure 49. Vertex 1 is the new vertex you defined, it will be on grid because you used the cursor to define it. However, vertices 2 and 3 are defined only by the intersections of the shifted sides and their connected sides. These vertices are not on the resolution grid, so the command will fail.

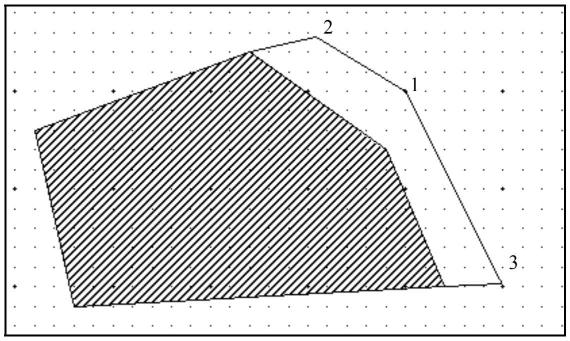


Figure 49: MOVE SIDE will fail to move the vertex selected in Figure 48.

ICED™ prevents you from accidentally using the MOVE SIDE command to move a vertex off of the resolution grid. To create a polygon like this you should start from scratch to create a new polygon so that all vertices will be on-grid.

Adding Circles

ICED™ supports polygon approximations for circles (also called round flashes). Special parameters are used in the ADD command to approximate the required circle as closely as possible. (You can also add polygons that approximate sectors and rings. Wire components can be created to approximate arcs. We will cover these components in following lessons.) The menu options to create these components are always available and are unaffected by the snap angle.

ICED™ approximates a circle with an *n*-sided polygon. Add a circle with 32 sides and a radius of 10 units using the following menu options:

1:**ADD** → (CIRCLE)**n=%** → (NSIDES)**32** → (RADIUS)**10.0**

The cursor is located in the center of the circle. However, once you press the left mouse button and define the center of the circle, this location is not stored as the location of a circular component. Instead the shape is stored as a polygon component, and the location of each vertex on the outside of the circle is stored in the definition. You can see this definition with the menu options:

1:(SELECT)**new**

2:(SHOW)**screen**

Note that the perimeter and area of the polygon are reported as well.

You could modify this component in any of the ways you can modify any polygon component. For example, create a box that shares an edge with the top of the circle and merge it with the circle with the menu options:

1:(**ADD)box**

2:(MERGE)**poly**

You can set the default number of sides for circular components with a USE command. To see what the default is now, use the menu options:

1:**USE → SHOW**

Your default is shown in the N_SIDES parameter. To change this value, use the menu options:

1:**USE** → (CIRCLE SIDES)**32**

Now you can add a circle with 32 sides with the menu options:

1:**ADD → CIRCLE** → (RADIUS)**10.0**

If you need to insure that all vertices are created on grid, you can set the resolution mode to HARD. Change this environment setting now with the following options:

3:**RESOLV** → (RESOLV STEP)**No Chg** → (RESOLV MODE)**hard**

When you add a circle, ICED™ may be forced to shift many of the vertices to be on grid. ICED™ enforces the rules that all vertices must be distinct and that no three vertices can be collinear. If you try to define a circle with a small radius and so many vertices that these restrictions cannot be met, the ADD CIRCLE command will fail with an error message. The recommended maximum number of sides is 32.

Let us see the warning message by adding a circle with 32 sides and a radius of 4 units.

1:**ADD** → **CIRCLE** → (RADIUS)**4.0**

Note that no component is created and that an error message is reported at the bottom of the window. You have two choices to create this circle:

- Decrease the number of sides (the best option)
- Change the resolution mode so that off-grid coordinates are valid for commands that calculate coordinates.

Use following menu options to change the resolution mode and create the same circle we were unable to create with the previous add command:

3:**RESOLV** → (RESOLV STEP)**No Chg** → (RESOLV MODE)**soft**

1:**ADD** → **CIRCLE** → (RADIUS)**4.0**

Let us look at the coordinates created to approximate this circle with the menu option:

2:(SHOW)**screen**

Note that virtually all of the coordinates are off of the resolution grid of 0.5 units. Let us assume that you always prefer to create your vertices on-grid and change the resolution mode back to hard with the menu option:

3:**RESOLV** → (RESOLV STEP)**No Chg** → (RESOLV MODE)**hard**

Adding Rings

Another type of circular component you can create is a ring. Let us suppose that you need to create a ring with an inner-radius of 4 and an outer radius of 8. Since the inner circle is so small, you should change the default number of sides back to a more reasonable value of 16 sides. This value will be used for both the inner and outer circles. Use the following menu options:

1:**USE** → (CIRCLE SIDES)**16**

1:**ADD** → **RING** → (INNER-RADIUS)**4.0** → (OUTER-RADIUS)**8.0**

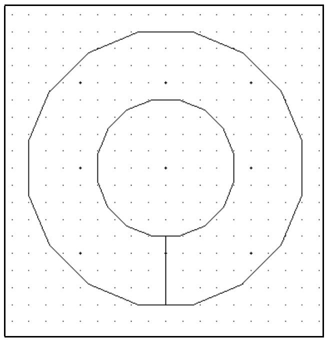


Figure 50: Ring component

Note the vertical line in the new component shown in Figure 50. This line really represents two sides linking the inner circle to the outer circle. ICED™ shapes cannot contain true holes. Instead this a simply a polygon shape folded around so that two sides touch.

If you do not want the same number of sides in the inner and outer circles, you can specify the number of sides for each circle with the menu. Let us assume that you require a shape that is a square 16 units wide with a circle 8 units wide cut out of the middle. You must remember to cut the dimensions in half because you specify radii rather than diameters in the ADD RING command.

1:**ADD** → (RING)**n=%,%** → (N_SIDES FOR INSIDE CIRCLE)**16** → **…**
…(N_SIDES FOR OUTER CIRCLE)**KeyBrd**→ (INNER-RADIUS)**4.0 …**
…→ (OUTER-RADIUS)**8.0**

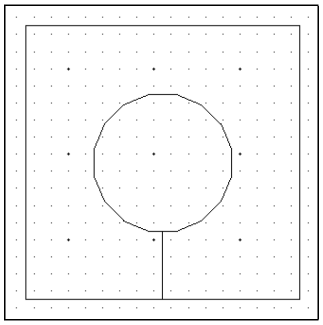


Figure 51: Ring with N_SIDES = 16, 4

Since the N_SIDES parameter for the outer circle is so small, it is not an option provided on the menu. You must type this value in using the keyboard. Type at the prompt:

4

The shape shown in Figure 51 is created. If you prefer to type commands, you could create the same component with the following typed command:

ADD RING N=16,4 R=4,8

Sector and Arc Components

Suppose you need only part of a circular shape. Sectors are portions of circles and arcs are portions of rings. You can create these with special ADD commands as well. Both are created specifying the beginning and ending angles, where angles are measured clockwise and 0º is at 12 o'clock. Negative angles are permitted.

Sectors are so rarely used that no menu entry is available. You must type the command on the command line. To create the sector shown in Figure 52, use the following command:

ADD SECTOR N=16 R=10 ANGLES=90,180

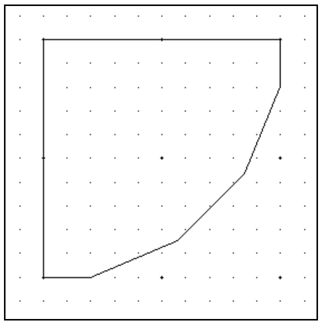


Figure 52: Sector with angles=90, 180

Actually the N=16 parameter is not required since this is the default number of sides we set with the USE command back on page 150. It was added to this command to make a point. Note that even though N_SIDES=16, the resulting sector does not have that many sides. The N_SIDES parameter controls how many sides are in an entire circle. When you create only part of a circle, it is created as though it was cut out of a circle with that many sides.

Arc components are not polygons. Instead they are wire components. Due to this, some of the parameters are handled a little differently than the other circular components. Since wire components have width, you either specify the width in the ADD command, or use the default width of the layer. (See page 116.) Wires have two different valid end types. Type 2 wires extend half of the wire width past the endpoints and type 0 wires end flat at the endpoints. The end type for arc wires is a separate parameter of the USE command. Let us set that value now by typing the following command:

USE ARC_TYPE=0

The method used to create the component given a specified radius is also slightly different. For arcs, the radius is measured from the center of the arc to the center of one of the wire segments. The arc extends past this radius for a little more than half of the width of the wire used to create it.

Now that we've confused you totally, we will create an arc using menu options and you will see that it is easier than you think. The menu even has special shortcuts for arcs that are ¼ of a circle and ½ of a circle. Let us create an arc using the default of N_SIDES=16 using the following menu options:

1:**ADD** → (ARC)**w=%** → (¼ CIRCLE)**90:180** → (WIDTH)**2.0** → (RADIUS)**10.0**

The arc will look like the one shown in Figure 53. Note that the radius is measured to the vertices of the wire component running down the center of the shape. Since the default wire type of 0 was used (set by the USE ARC_TYPE command on the previous page), the wire ends do not extend past the 90º sweep of the arc. Since a value of N_SIDES=16 (also set by the USE command) was used, the arc component is created as though it was cut out of a circular shape with 16 sides.

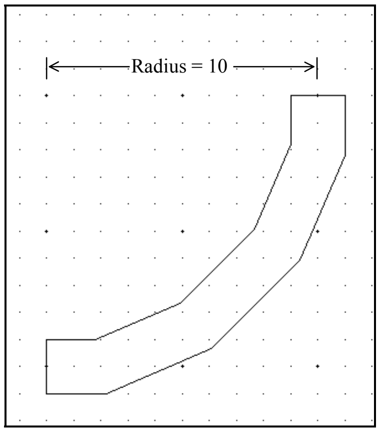


Figure 53: Arc component

Conclusion

If you will be adding circular components, we recommend that you read the more detailed explanation of these components given in the ADD command description of the Layout Editor Reference Manual.

If you want to know more about the resolution grid and its effect on creating components with non-perpendicular sides, read about the RESOLUTION command in the Layout Editor Reference Manual.

If you want to understand about the finest resolution you can use to define coordinates, read about the NDIV command line parameter in the same manual.

Exploring Cell Hierarchy

This tutorial focuses on ways to edit multiple cells (particularly nested cells) during a single edit session. Methods that allow you to get information about hierarchical designs and about components in deeply nested subcells will also be covered.

First a few definitions:

Root cell	the cell opened when you launch the layout editor
Parent cell	the cell you are currently editing when you execute a subcell edit command
Child cell	the cell you edit with a subcell edit command
Main cell	the highest-level cell of a design hierarchy

For this tutorial we will use a set of real design cells. You should delete or move the cells currently in the Q:\ICWIN[27]\TUTOR directory. To use DOS commands to clear the directory, open a console window with the ICED icon and type:

```
CD TUTOR
DEL *.* <Enter>
```

Now copy the all cell files in the Q:\ICWIN\SAMPLES\74181\CLEAN directory to the tutorial directory. The DOS command to accomplish this is:

```
COPY ..\SAMPLES\74181\CLEAN\*.*
```

Do not open the editor yet. First we must determine which cell is the main cell.

Determine Which Cell File Represents the Main Cell

From the list of copied cell files it is not immediately obvious which cell is the main cell. ICED™ provides a utility to aid you with this task, ICTOP.EXE. This utility will search the cell files in the current directory looking for a cell that is not nested in any other cell. (The search can be expanded to all cell libraries defined in the current ICED_PATH environment variable with the /f switch. See the Layout Editor Reference Manual.)

[27] Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software.

Type the following command in the console window:

ICTOP

The utility reports that the only cell that is a likely main cell is TOP181L.

List the Subcell Structure of a Cell

Another useful utility provided with ICED™ is the ICTREE.EXE utility. This utility displays the subcells nested in a given cell. To see how this works, type the following in the console window:

ICTREE TOP181L

The listing may be too long to fit in the console window, but you can save the information in a file with redirection. Type:

ICTREE TOP181L >ICTREE.TXT

NOTEPAD ICTREE.TXT

Note that the TOP181L cell contains references to only two other cells: TOP181 and VIA. The TOP181 cell is really the main cell in the design. The TOP181L cell is a dummy cell created to verify the TOP181 cell layout in an LVS comparison to a schematic file. Wires, vias, and text were added to the TOP181L cell to simulate some connections that will be made in a higher-level design cell. The highest-level design cell is really TOP181. The TOP181 cell contains several subcells. Note that B1STF is one of the subcells nested in the TOP181 cell.

```
Subcell directories:
  Path 0 = E:\ICWIN\TUTOR
  Path 1 = E:\ICWIN\SAMPLES

TOP181L(0)
  TOP181(0)
    B0STF(0)
      INV_1X(0)
      NAND_2(0)
      NAND_2S(0)
      NAND_3(0)
:
    B1STF(0)
:
  VIA(0)
```

Figure 54: Portion of ICTREE listing

Let us edit the TOP181L cell briefly to see what it looks like. First, **exit the file editor**, then launch the layout editor with the command:

ICWIN TOP181L

Note that all layers are drawn in yellow and the layout is very difficult to interpret. This is due to the fact that this cell was not created with a correct environment using the startup command file used for the other cells. Display the environment settings for this cell with the following menu option:

2:(TEMPLA)**screen**

Note that there are no layer definitions, and then return to the layout editor window by pressing both mouse buttons. If we use this cell as our root cell, this environment will be stored as the new environment for all cells edited during the editor session, overwriting the correct environment currently stored in the other cell files. **It is very important that the root cell have the correct environment you want stored in all subcell files you intend to edit during a layout session.** If all cell files are created using the same startup command file, then this is not an issue. However this is not the case with the TOP181L cell. It was created by another user who did not use the same startup command file.

The startup command file used to create the environment of the design cells in this example is not provided with the installation. We could create a command file to create the correct environment for the TOP181L cell by using the TEMPLATE command while editing one of the true design cells as the root cell. However, we will not do this here. (We will do this later in another tutorial on page 171.) Instead we can simply ignore the TOP181L cell and use the real top-level cell of the design, TOP181, as our root cell.

Quit the TOP181L cell without saving the cell file with the menu option:

1:**FILE → QUIT**

Now launch the editor to edit the TOP181 cell as the root cell with the command:

ICWIN TOP181

Let us view the entire cell with the menu option:

1:(VIEW)**all**

Selecting Nested Cells

The ICTOP listing we created earlier reported that one of the cells nested in TOP181 is the B1STF cell. How can we select this cell for further study? The SELECT command allows you to select a cell by name. The menu options even allow you to select from a list of subcells by name. Use the menu options:

1:**SELECT** → (CELL %)**all** → **B1STF**

The menu options under "CELL %" select cells with a specific name. The options under "CELL *" select cells without regard to their names.

Note that only 1 component was selected. This means that there is only one copy of B1STF in cell TOP181. Now that we have the cell selected, we can change the view window to zoom in on this cell with the menu option:

1:**VIEW** → **SELECT**

At this scale it is easier to see the select marks on the cell's bounding box. The bounding box is indicated by the dotted line surrounding the cell. Some segments of this bounding box are hidden by shapes.

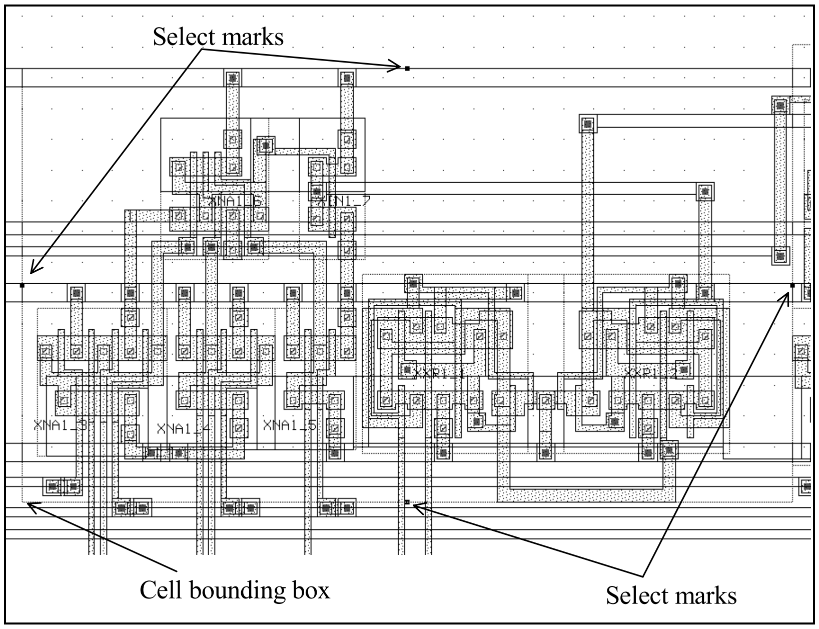


Figure 55: Zoomed in on B1STF cell

Reporting Nested Shape Information with (SHOW)@deep

Since you are unfamiliar with this design, and it uses different colors and patterns than the other cells we have used in previous tutorials, you may not be able to tell what layer a particular component is on by looking at the display. You also cannot determine which shapes are contained in which cells by looking at the display. We will demonstrate methods of editing nested cells in a moment, but if all you need is information on a nested component, you can obtain it with the (SHOW)@deep menu option. Select this option now.

2:(SHOW)**@deep**

The command file Q:\ICWIN\AUXIL\DEEPSHOW.CMD is now executing. The cursor changes to a near box with an 'X'. The command file is waiting for you to place the cursor on the edge of a nested component. Do this and click the left mouse button. (If you have trouble placing the cursor precisely at this scale, use the nested view menu to zoom the view window by pressing the <Esc> key then clicking on BOX to define the new corners of the view window.)

Once you select a component with the mouse, the view window is replaced with a report on the component. The top of the report indicates which cell contains the component. Type <Enter> or press both mouse buttons to return to the view window.

This concludes the DEEPSHOW.CMD command file. Note that the command file restores the selection status of components that were selected before the command file was executed. The cell B1STF is still selected.

Overview of Subcell Editing

There are three different subcell edit commands. We summarize the differences between the different commands on the next page. Basically, the EDIT command is similar to beginning a separate edit session to edit a different cell. The two other subcell edit commands are intended solely to modify subcells of the parent cell. With these commands, the coordinate system of the parent cell and the orientation of the child cell in the parent cell is maintained while you edit the child cell.

Command	**Which cells can be edited**	**Orientation**	**Coordinate system**	**Display of parent cell**
EDIT	Any cell in one of the existing cell libraries, or a new cell in the root cell directory	Orientation used to create the cell	Child cell i.e. the origin of the cell will be at (0,0)	No
T_EDIT	Only subcells of the current cell	Orientation in parent cell e.g. a rotated cell will remain rotated	Parent cell i.e. the origin of the cell will be at the location where it was placed in the parent cell	No
P_EDIT	Only subcells of the current cell	Orientation in parent cell	Parent cell	Yes

There are three methods used to select which cell to edit when you use a subcell edit command. These methods appear on the menus under the appropriate subcell edit command heading. These methods are summarized below. You will see examples of these methods in the following lessons.

Cell	**Select**	**Near**
Available only in EDIT command. Choose cell by name from a list of open cells or from lists of cells in each cell library.	Open selected cell if one is already selected. Will allow you to select a cell if none are currently selected. Travels down only one level of hierarchy.	Open cell that contains the component you indicate with the near-cursor. Travels through cell hierarchy to the level of indicated component.

All of the subcell edit commands suspend the editing of the parent cell. Other commands executed after a successful subcell edit command affect only the child cell. You must return from editing the child cell with a LEAVE, EXIT, or QUIT command to continue to edit the parent cell. (We describe these options for returning to the parent cell on page 160.)

Using T_EDIT

T_EDIT stands for Transformed Edit. This command is used to edit a cell in the transformed coordinate system of the parent cell. In other words, you can edit a nested cell without changing the view window or the location of the origin (the point 0,0.) Even if the coordinate system is not important to you, this command is useful for maintaining the view of the layout as you travel into nested cells. Cells that are rotated or mirrored will retain that orientation as you edit them.

This command is particularly useful for going into nested cells one level at a time. Let us edit the selected cell with the menu options:

2:**EDIT →** (T_EDIT)**select**

Now only the B1STF cell is displayed, but it has not changed location on the screen. The components of this cell can now be modified. ADD commands will add components to this cell, not the root cell.

Look at the top of the window. Note that the following text is reported:

TOP181:B1STF

The cell name on the far left is the name of the root cell. The name of the cell you are currently editing is on the right. As we go further into the cell hierarchy, this list will grow longer.

Let us go one level deeper into the cell hierarchy with the T_EDIT command again. This time we will not select the cell in advance.

2:**EDIT →** (T_EDIT)**select**

Now place the cursor over the ‘5’ in the text “XNA1_5” near the center of the window and click the left mouse button. Now the text at the top of the window reads:

TOP181:B1STF:NAND_2

You are now editing the cell whose bounding box surrounds the location of the cursor. In this case it is the cell NAND_2.

Returning to the Parent Cell

There are several options for returning from editing a subcell. These are all available from the 1:**FILE** menu option, or by typing them at the command prompt.

EXIT	Use this option when you definitely want to overwrite the cell file with the modified child cell when you close the editor.
QUIT	Use this option when you definitely **do not** want changes just made to the child cell saved in its cell file when you close the editor. All changes to the child cell are lost when you return to the parent cell. When you use this option to quit the root cell and close the editor, only changes made to the root cell are lost. Changes made to child cells will still be saved in cell files if you did not use the QUIT command to return from editing those cells.
LEAVE	Use this option when you want to save a modified child cell file, but leave an unmodified child cell intact with its previous date stamp.
JOURNAL	Use this option to terminate the editor immediately without saving any cell files.

The LEAVE command is the most useful way of returning from a subcell edit. This allows you to return from subcell edits with the same command each time without thinking about whether or not to save changes. Type this command twice now to return to the root cell, TOP181. The cell files will not be overwritten in this case since you made no changes to the cells. (You can use the 1:**Again** menu option or the < ↑ > key to repeat the command.)

LEAVE

LEAVE

Editing a Cell in Place

If you prefer to leave the parent cell displayed on the screen for reference as you edit a subcell, use the P_EDIT command. This command operates in a very similar manner to T_EDIT. It is most useful for traveling down several levels of cell hierarchy and editing nested components almost as easily as if the components were nested in the parent cell.

Select the following menu option:

2:**EDIT** → (P_EDIT)**near**

Each **near** option under the EDIT, P_EDIT, and T_EDIT headings allows you to travel all the way down to the level of the cell that contains a specific component. You must place the near-box cursor on the edge of a nested component, then when you click the left mouse button, you are editing the cell that contains that component.

Move the cursor now over an edge of a component just above the text "XNA1_5". (You may need to change the display scale with the nested view menu using the <Esc> key.) Be sure that the cursor is not in the middle of a component or on the white dotted outline of the cell. Click the left mouse button.

Note that the message at the top of the window reports that you are now editing the NAND_2 cell. At first, it looks as though nothing has changed. But if you look closely, you can see that only components in the NAND_2 cell are drawn with fill patterns. Only these components can now be edited.

Try to use any of the select commands to edit any component drawn in outline mode. Nothing will be selected. You can modify only components in the selected copy of the NAND_2 cell. If you add components, they are added to this subcell, not the parent cell. Let us test this by adding a big text component with the menu options:

1:**ADD →** (TEXT)(vert)**s=% → 20.0**

Type something like "I did it" at the prompt, press <Enter>, place the component over the cell, and then press the left mouse button. Now display the whole parent cell with the menu option:

1:(VIEW)**all**

Note that every copy of NAND_2 in the drawing has been instantly modified. We don't really want to save this kind of change, so return to the parent cell without saving changes to the child cell with the menu option:

1:**FILE** → **QUIT**

One Way to Modify Only One Copy of a Cell

Suppose that you need to modify only one specific copy of the NAND_2 cell. There are many methods to accomplish this. In this lesson we will use the

UNGROUP/GROUP method. This method is appropriate for simple edits. We will cover a method using the SWAP command later in this tutorial that is preferable when you need to make more extensive changes.

To use this method, you need to edit the cell that contains the cell that requires modification. In this case, the B1STF cell. This cell should still be selected in our root cell. (If it is not, use the command on page 156 to select it again.) Now edit this cell with the SELECT option of the P_EDIT command with the menu options:

2:**EDIT** → (P_EDIT)**select**

Now select the NAND_2 cell we have edited before with the options:

1:**SELECT →** (CELL %)**all → NAND_2**

Note that only one cell is selected since there is only one copy of NAND_2 in cell B1STF. (If there were more than one copy of NAND_2, we should have used the (CELL %)**in** option to select only one copy.)

Now we will ungroup this cell, change one component, and regroup the components into a new cell with a new name. First, ungroup the cell with the menu options:

2:**EDIT → UnGRP**

Note that now no components are selected. We need to save all of the components that make up this NAND circuit into a set of selected components so we can regroup them later into a new cell. To select the components just ungrouped we use the menu option:

1:(SELECT)**new**

To save a set of selected components, use the select stack feature. Use the menu options:

1:**SELECT → PUSH**

Now unselect everything and then select a side of a single component to modify with the menu options:

1:(UNSEL)**all**

1:(SELECT)**side**

You can select a side of any component in the NAND circuit for modification. Make a big, noticeable change to this component with the menu option:

1:(MOVE)**side → X&Y**

Now reselect all of the components that make up the NAND circuit by using the select stack with the menu options:

1:**SELECT → POP**

Regroup the components into a new cell with the options:

2:**EDIT** → (GROUP)**at ***

This option creates the origin of the new cell at the lower left corner of a bounding box that just surrounds all of the components. Type the following name at the prompt for the new cell name:

NANDX

Now a cell with the name NANDX has replaced the NAND_2 cell. None of the other copies of NAND_2 in the design are affected.

Return to the root cell with the LEAVE option so that we save this change.

LEAVE

The EDIT SELECT Command

Both the T_EDIT and P_EDIT commands allow you to edit subcells in the orientation they use in the parent cell. When this is undesirable, you can use the EDIT command instead. In this case, you edit the cell almost as though you opened it in a new layout editor session.

The only real difference is that the environment of the root cell is still used. The environment of the root cell is the only environment database used for all cells edited in a layout editor session.

Let us edit a different cell this time. First select the only copy of the NAND_4 cell in TOP181 with the options:

1:**SELECT →** (CELL %)**all → NAND_4**

Now change the view window to zoom in on this cell with the options:

1:**VIEW → SELECT**

Note the orientation of this circuit. Now use the EDIT command to open the cell with the menu options:

2:**EDIT →** (EDIT)**select**

Note that the orientation of the cell in TOP181 is ignored. Make a small change to this cell if desired and then return to the root cell with the LEAVE command.

1:**FILE → LEAVE**

Unselect the cell and redisplay the entire root cell with the menu options:

1:(UNSEL)**all**

1:(VIEW)**all**

Using EDIT to Create a New Cell

The EDIT command is the only subcell edit command that can be used to create or modify a cell that is not nested in the hierarchy of the root cell. Let us create a new cell now with this command. Use the menu options:

2:**EDIT →** (EDIT)**cell → KeyBoard**

If you selected a cell name from the list instead of "KeyBoard" then you could choose from the list of subcells in the root cell. Using "NextPATH" repeatedly will display lists of existing cells in every cell library.

When you do use "KeyBoard", you type the cell name. If a cell with that name already exists in the root cell, or in any of the cell libraries, you will edit that existing cell. If the cell is not found in any cell libraries, then you will create a new cell file in the directory of the root cell. (The cell file is saved when you terminate the editor.)

Type in the following new name at the prompt:

MYNEWCELL

You are now looking at a blank view window. Where is the display grid? When you changed the view window to display the entire cell in the previous lesson, you changed the display scale to so large a scale that all grids were too fine to be displayed. You must zoom in to change the view scale to a reasonable value for creating geometry to see the display grids. Do this with the menu options:

1:(VIEW)**in % → 10.0**

Create some geometry if you desire and return to the root cell with:

1:**FILE → LEAVE**

Another Method to Modify Only One Copy of a Cell

Another way to replace a single copy of a cell with a modified version is to create a new empty cell, copy the contents of the old cell into the new cell, swap the old cell with the new one, and finally edit the new cell in place.

Create the new cell with the menu options:

2:**EDIT →** (EDIT)**cell → KeyBoard**

Type the cell name at the prompt:

NAND_4_MOD

Now you are editing a new blank cell. Add the NAND_4 circuit to this cell.

1:(ADD)**cell → NAND_4**

Carefully use the mouse to place this cell at the coordinates (0,0). Look at the report of the coordinates of the cursor at the bottom left of the window.

Select the cell and use the UNGROUP command to explode this cell into its components.

1:(SELECT)**new**

2:**EDIT → UnGRP**

Now return to the root cell.

1:**FILE → LEAVE**

You have now created an exact copy of the NAND_4 cell. The effect is the same as copying the cell file outside of the layout editor.

Now that we are back in the root cell, we can swap the new cell for the original cell. Since we will use the SWAP command for this, we must be careful to select only the single copy of the cell we wish to swap. If the number of components selected is not 0 (as indicated in the command prompt) unselect all components with 1:(UNSEL)**all**. Now select only the NAND_4 cell and zoom the view window to display it with the menu options:

> 1:**SELECT →** (CELL %)**all → NAND_4**
> 1:**VIEW → SELECT**

Swap the modified cell for the existing copy of NAND_4 with the menu options:

> 2:(SWAP)**cells → NAND_4 → NAND_4_MOD**

The cell remains selected. Now edit this cell in place.

> 2:**EDIT →** (PEDIT)**select**

You can see that you did indeed swap the cell successfully and that you are now editing your new copy by noting the name of the cell you are editing at the top of the window. Modify this cell by adding some text to it with the menu options:

> 1:**UseLay → TXT →** (TEXT)**hori**

Type in a string like MODIFIED and place the text component with the mouse. Then return to the root cell with the menu options:

> 1:**FILE → LEAVE**

The only change you can see is that the text component added to the new version of the circuit is now displayed. No other copies of NAND_4 in the design are affected.

(Always be careful to note what components are selected when you use a SWAP command. If multiple copies of NAND_4 were selected when the SWAP command was executed, all copies would have been swapped.)

Editing Cells in Protected Libraries

The batch file you used to open the layout editor specified the Q:\ICWIN\SAMPLES directory as a cell library. This directory was given the default protection status; i.e. it is a view-only cell library. (See page 64 for more details on cell libraries.) You are not allowed to edit the cells in this library. When you select such a cell for editing, you cannot save any changes.

To see how this works, let us try to edit a cell in the SAMPLES directory with the menu options:

2:**EDIT →** (EDIT)**cell → NextPATH → NextPATH → NAND**

You will probably hear a beep and see a warning message similar to:

"NAND already exists in the view-only directory Q:\ICWIN\SAMPLES\
Enter option (V=VIEW only; C=CANCEL):"

ICED™ is warning you that you are attempting to edit a cell in a protected cell library. This is to prevent you from editing cells that are shared among other designers. Respond by typing a **<V>** to proceed.

The edit commands are not disabled. The menus look exactly the same as before. However you will not be able to execute the EXIT command to mark this cell for saving. Try this now with the menu option:

1:**FILE → EXIT**

After viewing the error message, return to the root cell with the following menu option:

1:**FILE → QUIT**

If the SAMPLES directory had been given copy-edit protection instead of view-only protection, the warning prompt issued by the EDIT command would have been different.

"NAND already exists in Q:\ICWIN\SAMPLES\
Enter option (V=VIEW only; L=>LOCAL copy; C=CANCEL):"

If you reply with a <V>, then the EXIT command will be disabled as we saw above. If you reply with a <L> instead, then you will be able to modify the cell and save the changes, but the modified cell file will be saved in the root cell directory (Q:\ICWIN\TUTOR) rather than in the SAMPLES directory.

Conclusion

This concludes the tutorial. If you will be continuing with other tutorials, you should leave the cell files in the TUTOR directory. You will be using this design in the next tutorial.

If you will not be continuing with the next tutorial immediately, you should close the editor now with the menu option:

1:**FILE → LEAVE**

Manipulating the Display

This tutorial focuses on commands that change how components are displayed in the view window. This includes information on how to define the appearance of individual layers. Other commands that control the appearance of the view window will be covered as well.

You can use the same cell files we used in the last tutorial. If you do not have the TOP181 cell files in the TUTOR directory, refer to page 153 for instructions on how to copy them into the directory.

Open a console window and open the layout editor to edit the TOP181 cell.

```
ICWIN TOP181
```

Setting the Properties of a Layer

The LAYER command is used to set the properties of each layer. You would usually set the properties of all layers in the startup command file rather than using commands in the editor to modify these properties in individual cells. Use of a startup command file makes these environment settings consistent for all cells in a design. However, there may be occasions when you want to change the appearance of extra layers (e.g. DRC error layers) or temporarily change the appearance of design layers to help you see some layout configurations more clearly.

The properties of layers you set with the LAYER command (or the 3:LAYER menu entry) are as follows:

Keyword	**Property**	**Covered in examples on pages**
Name	Defines a string that can be used in place of the layer number to identify the layer	116
CIF	Defines a string that will be used to identify the layer in CIF export	-
STREAM	Defines layer number and data type that will be used to identify the layer in STREAM export	214

WIDTH	Sets default width of new wires added to the layer	63
SPACE	This value is used to define the distance between guide marks of the spacing cursor when adding new components on the layer	121
COLOR	Sets the color used to draw the layer	114
PATTERN	Sets the fill pattern used to draw the layer in the display	173
PEN	Sets the fill pattern used to draw the layer in plots	191

Figure 56: Layer properties

Let us set all of these properties for an unused layer number using menu options.

3:**LAYER** → **By #** → **KeyBrd** → **SetALL** → (LAYER NAME)**KeyBrd** → (CIF NAME)**KeyBrd** → (STREAM LAYER)**10** → (DATA TYPE)**10** → (WIDTH)**3.000** → (SPACE)**2.000** → (COLOR)**4:** → (PAT'RN)**14:** → (PEN)**pen=***

At the prompt "Enter layer name, number, or *" , type the layer number below. This first prompt identifies which layer to modify, and since we are modifying a layer that does not yet have a name defined for it, we must use the number to identify it. Type the following:

100

At the prompt for the name, type:

MYLAYER

At the prompt for the CIF name, type:

MYL

Since you specified PEN=*, the pattern used for plotting will be the same one used for the display.

The history line reports the new layer settings. If you want to see these settings again at a later time, you can type the LAYER command with the layer name or number as the only parameter. This will not only report the settings, but will make that layer the default layer. Note that the last command did not change the default layer by looking at the command prompt where the default layer is listed. Now display the current settings and change the default layer by typing:

LAYER MYLAYER

Change the view window to an empty space and add a component on this layer to see how your settings affected its appearance. Use one of the 1:(ADD) options.

Displaying and Exporting Layer Properties

You can see a list of all layer properties with the TEMPLATE command. This command is used to report not only layer properties, but also all other environment settings. To list all environment settings on the display, use the menu option:

2:(TEMPLA)**screen**

Use the scroll bar on the right side of the window to scroll the display if necessary to display the settings for all layers near the bottom of the report. Note that only named layers are listed. Note also that the list begins with the following line:

```
LAYER * WIDTH=0.1 SPACE=0.0 MAGENTA PAT=0 PEN=1
```

The layer * settings are used as the defaults for all unnamed layers. We will explore layer * more in the next lesson.

The template report can be exported to a command file. If you did the previous tutorial (Exploring Cell Hierarchy beginning on page 153,) then you may remember that the main cell in our example, TOP181L, did not have the environment settings (including layer colors) used in the other design cells. We will now export the environment settings from the TOP181 cell and import them into the TOP181L cell. This will result in the same layer properties in both cells.

First export the environment settings from the current cell (TOP181) with the menu option:

2:(TEMPLA)**file**

At the prompt, type the following string for the base of the file name:

TOP181ENV

Note that the history line reports that the file "TOP181ENV.CMD" has been created. This file has been created in the current directory. If you needed to, you could edit this file with any ASCII editor to add, delete, or remove any of the commands that control the environment settings. We will not do this at this time.

Now close the layout editor with the EXIT command. If you have changed only environment settings in this cell, the LEAVE command will not save the cell file. Always use EXIT when you are changing environment settings.

1:**FILE → EXIT**

Open the TOP181L cell by typing the following in the console window:

ICWIN TOP181L

Now execute the command file you created above with the menu option that allows you to select a command file to execute:

3:**@%.cmd → TOP181ENV**

Now that the layer properties have been defined, the layout is drawn in recognizable colors. Since we changed only environment settings we need to close this cell with the EXIT command to save those changes.

1:**FILE → EXIT**

Reopen the TOP181 cell to use for the rest of the tutorial.

ICWIN TOP181

*Layer **

The properties of layer * set the defaults used for all unnamed layers. The startup command file supplied with the installation (Q:\ICWIN\SAMPLES\NEW.CMD) contains the following line to set the properties of layer *:

```
LAYER * WIDTH=2.0  SPACE=0.0  YELLOW  PAT=0  NO_PEN
```

When you add components to a layer number that has no name assigned to it, the line above indicates that the components will be drawn in yellow with no fill pattern and they will not be plotted at all.

The creators of the TOP181 cell preferred that components on unnamed layers be drawn in magenta on both the display and in plots. So they changed the LAYER * line in their startup command to:

```
LAYER * WIDTH=0.1  SPACE=0.0 MAGENTA PAT=0 PEN=1
```

It is rare to use this command outside of a startup command file. However you can change the settings of layer * by using the following menu option:

3:**LAYER → * → SetALL**

Fill Patterns

The fill pattern used to draw each layer is assigned by the LAYER command. Let us change the fill pattern used for the MYLAYER layer with the menu options:

3:**LAYER → MYLAYER → pat'rn →** (PATTERN)**20:**

If you prefer that components on a given layer are always drawn in outline mode, you can assign pattern 0. If you want a solid fill pattern, assign pattern 1.

Whether or not all layers are drawn with their assigned fill patterns depends on the fill mode. Toggle the fill mode now with the menu option

3:(FILL)**tog**

Now toggle it back on with the menu option:

1:**Again**

The patterns available in this cell file are the patterns supplied with the installation. The definitions are stored in the file Q:\ICWIN[28]\AUXIL\SAMPLE.STI. "STI" stands for "stipple" patterns. This is not an ASCII file; you cannot edit it directly.

If you want to create new patterns or modify the patterns supplied with the installation, you must follow this process:

- Copy the source file for the sample patterns, Q:\ICWIN\SAMPLES\SAMPLE.DAT.
- Edit this file to change the patterns as desired.
- Compile the file to the .STI format with the MkSTI.EXE utility supplied with the installation.
- Move the .STI file to the Q:\ICWIN\AUXIL directory.

[28]Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software.

- Change the environment setting that controls which pattern file is loaded with the PATTERN command or the 1:FILE → PATTERN menu option

We will not perform this process now. Few users feel the need to create new patterns since the 36 patterns supplied with the installation are almost always sufficient. However, if you want to create custom patterns, read the description of the syntax of the pattern source file in the MkSti utility description in the Layout Editor Reference Manual.

The COLOR Command

This command is used to define the color used for each of the 16 color numbers. Each color number has three color definitions, all set by this command.

Keyword	**Use**	**Format**
Palette	Define the color used in the display	(*red_value, green_value, blue_value*)
ONE_DOT	Define one of the 8 supported raster printer colors for plotting in DOTS=1 mode	One of the keywords: BLACK WHITE YELLOW GREEN RED CYAN BLUE MAGENTA
FOUR_DOTS	Define a pattern of 4 dots of color for each pixel of the layout when plotting in DOTS=4 mode	(*dot_color1, dot_color2, dot_color3, dot_color4*) where each dot color is one of the colors listed above.

Figure 57: ICED color definitions

We will explore the ONE_DOT and FOUR_DOTS parameters in the Plotting tutorial on page 194.

In addition to the parameters above there are two other keywords. The NAME keyword is used to assign a name to color numbers in the range 8:15. You cannot change the names of colors 0:7. The LEVEL keyword controls how components are drawn when different colors overlap. The color with the higher level number covers a color with a lower level number. Areas where components with colors at the same level overlap are drawn with pixels that alternate between the colors. Any time 5 or more colors overlap, the overlap area is drawn in white on the screen display and in black on plots.

Since color definitions are almost always defined in the startup command file, there is no menu entry for the COLOR command. You must type these commands in a command file, or at the command prompt. Note the values set by the startup command file supplied with the installation in Figure 54. (The ONE_DOT and FOUR_DOTS parameters are not listed. We will discuss those settings in the Plotting tutorial.)

```
COLOR  0   NAME=BLACK        PALETTE=( 0, 0, 0)
COLOR  1   NAME=WHITE        PALETTE=(63,63,63)    LEVEL=16
COLOR  2   NAME=YELLOW       PALETTE=(63,63, 0)    LEVEL= 6
COLOR  3   NAME=GREEN        PALETTE=(21,63, 0)    LEVEL= 6
COLOR  4   NAME=RED          PALETTE=(63, 0,21)    LEVEL= 8
COLOR  5   NAME=CYAN         PALETTE=( 0,42,42)    LEVEL= 9
COLOR  6   NAME=BLUE         PALETTE=( 0, 0,63)    LEVEL=10
COLOR  7   NAME=MAGENTA      PALETTE=(63, 0,63)    LEVEL= 8
COLOR  8   NAME=GRAY         PALETTE=(42,42,42)    LEVEL=14
COLOR  9   NAME=BROWN        PALETTE=(32,16, 0)    LEVEL= 8
COLOR 10   NAME=ORANGE       PALETTE=(63,31, 0)    LEVEL= 8
COLOR 11   NAME=PURPLE       PALETTE=(21, 0,14)    LEVEL= 3
COLOR 12   NAME=DIM_RED      PALETTE=(22, 0, 0)    LEVEL= 3
COLOR 13   NAME=DIM_BLUE     PALETTE=( 0, 0,22)    LEVEL= 3
COLOR 14   NAME=DIM_GREEN    PALETTE=( 0,22, 0)    LEVEL= 3
COLOR 15   NAME=HI           PALETTE=(63,63,63)    LEVEL=15
```

Figure 58: Color definitions in NEW.CMD

You are not allowed to change the color names assigned to colors 0:7, but you can change their display palette, level, and four_dots parameters. Let us change the palette value of the color BLACK, color number 0. This color is used to draw the background of the view window. Type the following at the command prompt:

COLOR BLACK PAL=0,0,10

Now we will experiment with the settings for the color DIM_BLUE, color number 13. First let us assign this color to the layer MYLAYER that we created earlier in this tutorial. (If you have changed the view window so that the component you created on this layer is no longer visible, change the view window so that it is displayed.) Change the color of this layer with the menu option:

3:**LAYER → MYLAYER → color →** (COLOR)**13: →** (PAT'RN)**NoChg**

Note that the component is almost invisible against the dark blue background. Display the current parameters for this color by typing:

COLOR 13

Note that the history line reports the current settings for this color. The palette values are set to (0,0,22). This is very close to the values we used for the background color above. Change the palette values for color 13 to make it slightly more visible with command:

COLOR 13 PAL=(10,10,30)

If you need to change color values for your design, please read the additional information on the COLOR command in the Layout Editor Reference Manual.

Blanking and Unblanking Layers

When you are looking at the layout for a design, it can be difficult to focus on the layers you are interested in if there are many other layers drawn on top of them. You can turn off the display of some layers so they are not visible to make the layout easier to follow. The command to accomplish this is the BLANK command.

Before we execute a BLANK command, change the view window so it shows a fairly dense area of the TOP181 design. Type the following command to display a specific area of the layout:

VIEW BOX (0,0) (50,50)

Now turn off the display of the MT2 and TXT layers with the menu options:

3:**BLANK → layers → MT2 → Nxt Pg → TXT → Return**

The list of layers that is blanked is stored in the cell file. You could close this cell and reopen it at a later time and these layers would still be blanked.

To make the layers visible again use the menu options:

3:**UNBLANK → all**

Let us re-blank the layers by undoing the UNBLANK command with the menu option:

1:**UNDO**

Now suppose that you need to edit a subcell of the TOP181 cell. Edit a subcell with the menu option:

2:**EDIT →** (T_EDIT)**select**

Place the cursor over a component and press the left mouse button to begin editing a subcell. (Almost all of the components you can see with this view window are nested in subcells.)

Now that you are editing a subcell, note that MT2 wires and text on layer TXT are now visible again. Since this is a different cell, its list of blanked layers is separate from that in the root cell. However it is easy to blank the same layers in this cell. Just type the BLANK command with no parameters.

BLANK

Now return to the root cell without saving this change to the subcell with:

1:**FILE → QUIT**

You can use the BLANK menu options to easily blank all layers except for a few specific layers. Let us unblank all layers, then blank all layers *except* for MT1 with the following options:

3:**UNBLNK → all**

3:**BLANK → layers → MT1 → Invert → Return**

As you can see, all layers except MT1 are blanked in the drawing. The dotted subcell outlines are still drawn. (We will explore the display of subcell outlines next.)

New components are always added to a drawing with their blank flag cleared. Add a new component to the drawing on one of the blanked layers with the menu options:

1:**UseLay→ MYLAYER → BOX**

To reapply the list of blanked layers and therefore blank our new box, use the BLANK command with no parameters as before:

BLANK

Note that the new box is now blanked. Now unblank all layers.

3:**UNBLNK → all**

You may have noticed the options **root** and **cell** on the BLANK menu. The root option blanks only layers in the current cell. Layers in subcells are still drawn. Conversely, the cell keyword blanks only layers contained in subcells. When you do not use either of these options, the BLANK command applies to both components in the current cell and its subcells.

Let us suppose that you need to see what components are located in cell TOP181 as opposed to components in subcells. Use the menu options:

3:**BLANK → cell → SetAll → Return**

Now it is easy to see what components are at this highest level of the design.

Subcell Outlines

The last BLANK command turned off the display of the layers in subcells, but the cell outlines drawn with dotted white lines still remain on the screen. The display of these cell outlines is controlled with the OUTLINE command. You set the maximum depth for which cell outlines will be drawn, where depth refers to how many levels down a cell is nested.

Turn off the display of all cell outlines, except for subcells that have a depth of 1. This means that only subcells that are nested directly inside of the TOP181 cell will have their outlines displayed.

3:**OUTLIN → 1**

Before we go on, turn the display of all layers back on with the menu options:

3:**UNBLNK → all**

Blanking Specific Components

You do not have to blank entire layers. You can turn off the display of individual components with the BLANK command as well. Let us use this feature to turn off the display of only one or two subcells. First we will select only the component(s) to blank.

1:(UNSEL)**all**

1:**SELECT →** (CELL *)**in**

(Remember that the "CELL *" options select cells with any name. The "CELL %" options select only cells with specific names.) Draw the selection box to cross at least one of the subcells in the design. Now blank just that cell (or cells) with the menu option:

3:**BLANK → select**

Note that this method of blanking subcells also turns off the display of all cell outlines. Turn the display of all components on all layers back on with the menu options:

3:**UNBLNK → all**

Panning the View Window

There are two methods to pan the view window. Both are subject to the current cell's environment settings.

Method	**Available when**	**Menu option to set mode**
Arrow keys	Anytime	3:**ARROW → pan** Arrow keys will pan window. (In this mode, you must press the <Ctrl> key to use the arrow keys to navigate the command history.) 3:**ARROW → edit** Arrow keys will pan window only when <Ctrl> key is pressed at the same time. (Pressing the arrow keys in this mode without the <Ctrl> key will navigate the command history.)
Autopan	While digitizing coordinates	3:**A-PAN** Toggles autopan mode on or off. When autopan mode is on, the window will pan in the direction of the cursor when the cursor moves beyond the edge of the view window.

The Arrow mode should be currently set to edit mode. Pan the view window now by pressing and holding the <Ctrl> key and typing the arrow keys. Now release the <Ctrl> key and press the < ↑ > key. Now the arrow key recalls to the command line the last command executed.

The autopan mode is currently off. Turn it on now with the menu option

3:**A-PAN**

Now begin to use the RULER command to test this feature.

2:**RULER**

Click once anywhere in the view window, then move the cursor just past the edge of the view window. The view window pans ½ of the window dimension to follow the cursor. Click again anywhere to complete the command.

You can control how sensitive the cursor is to the edge of the window with parameters of the AUTOPAN command. Refer to the Layout Editor Reference Manual for details.

Some users find that autopan is distracting. You may prefer to leave it off. Users also have preferences about the arrow mode. If you are more comfortable with one mode or the other, you may change it whenever necessary with the 3:**ARROW** menu entry. The setting you choose will be stored in the cell file.

Nested View Commands and Using the Spacing Cursor

When you are in the process of executing a command, sometimes simply panning the window is not enough to change the view window to what you require. The nested view menu allows you to zoom the view window in or out, or pan the window by redefining the center of the new view window. You can use the nested view menu repeatedly during the execution of a command.

You activate the nested view menu with the center button of a three-button mouse, or with the <Esc> key. Use either method during the following exercise.

You will add a wire, changing the view window several times while digitizing coordinates. First, display the entire design with the regular view menu.

1:(VIEW)**all**

Let us use the spacing cursor for this exercise. Turn the spacing cursor on and set the guide marks to indicate a minimum 0.5 distance between this component and other MT2 components. Use the menu options:

3:**SPACER** → (SET)**space** → **0.500**

3:**SPACER** → (ON/OFF)**on**

Now begin to add a wire on the MT2 layer with the following menu options:

3:**UseLay** → **MT2** → (WIRE)**w=%** → **1.000**

Before digitizing the first coordinate, change the view window with the nested view command. Press the <Esc> key or center mouse button. We will indicate menu options from this menu by using "NVM" (for nested view menu) instead of a top-level menu number.

NVM:**BOX**

Now zoom in around the lower left corner of the design. Repeat this process if necessary until you can clearly see the left ends of the wires labeled S2 and S3.

Digitize the first coordinate of the wire midway between the S2 and S3 wires as shown in Figure 59. Now add more corners to the wire as shown in Figure 60. Use the spacing cursor to help you jog the wire through the narrow gap between wire S2 and the MT2 box in the via structure.

Now return to the view window in use before you began adding the wire with the nested view option:

NVM:**HOME**

At this scale you can just see the next via structure obstruction to your new wire as it continues to the right. Use the nested view menu again to zoom in on this

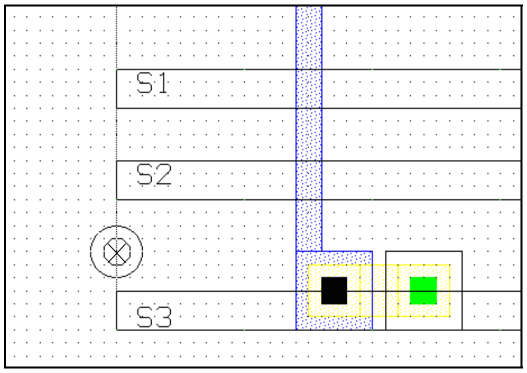


Figure 59: First coordinate of new wire

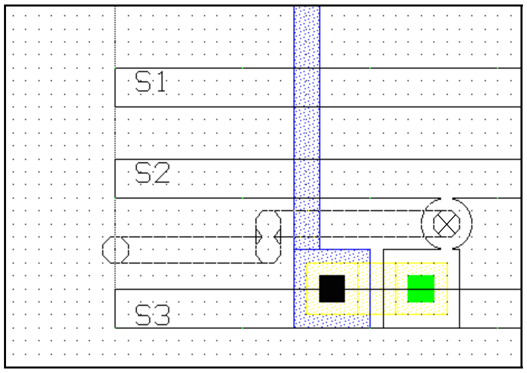


Figure 60: Continuing to digitize wire

area to carefully digitize a jog in the wire through the gap between the S3 wire and the via structure.

NVM:**BOX**

Continue to use nested view menu options to add more jogs to the wire until you feel comfortable with changing the view window while adding a new component. After you have added the final coordinate, press the right mouse button to complete the wire.

If you have not left the final view window displaying the entire design, change the view window now with the option:

1:(VIEW)**all**

Simplifying the View with VIEW LIMIT

When you display the entire contents of a large chip, there can be a noticeable delay in how long it takes ICED™ to draw the screen. Since you usually display a design at this scale only to zoom in on a specific area, there is no reason to display all components in complete detail. The view limit feature of ICED™ can automatically limit detail at fine scales so that the display draws instantly. Drawing less detail at these extreme scales actually makes the view of the design less cluttered and it is easier to zoom in on the area of interest. After you zoom in, the view limit automatically becomes inactive and the design is drawn in complete detail.

There are several parameters of the VIEW LIMIT command. You can set the scale at which the view limit becomes active. You can explicitly turn the feature off, so that the view limit never becomes active. You can limit the minimum size of individual components that are drawn when the limit is active.

The most useful parameters of the view limit feature are the depth for which cells will be drawn and the list of layers that will be displayed. The cell depth parameter has the biggest effect in how long it takes to display a large design at a fine scale. Cells nested more deeply than the indicated depth will be ignored entirely, speeding up display time dramatically.

Our sample design is too small to have a noticeable display delay, but we can still show the effects of the view limit. First let us change the view limit cell depth to a small number with the following options:

1:**VIEW** → (LIMIT)**set** → (SET)**depth** → (DEPTH)**0**

The command history line should report all of the current settings of the VIEW LIMIT feature. (If you experimented first and turned off the view limit, turn it back on with the option 1:**VIEW** → (LIMIT)**on**.) Now only components nested directly in the main cell will be displayed when the view limit becomes active.

To make the view limit active, you need to zoom out a few times. The exact zoom factor depends on the size of your window and your screen resolution. Zoom out once with the following menu option:

1:(VIEW)**out %** → **1.5**

If you do not see the display change so that the contents of many cells are no longer displayed, use the following option repeatedly until the cells disappear:

1:**Again**

Now that the view limit is active, let us refine the display further by limiting which layers are displayed while the view limit is active.

1:**VIEW** → (LIMIT)**set** → (SET)**layers** → **PWEL** → **MT2** → **Return**

Now only the PWEL and MT2 layers are displayed along with cell outlines. (Note that names of the cells nested directly in the main cell are displayed inside their bounding boxes. This is enabled by the 3:**DISPLAY** → **Cell Labels** setting. If you have zoomed out too far, the cell labels will not be displayed. In this case, try zooming in slightly a few times with 1:(VIEW)**in %** → **1.1**.)

Now zoom in on one cell with the menu option:

1:(VIEW)**box**

Note that all nested cells and layers are displayed now that the view limit is no longer active.

Limiting the Display of Arrays

Another way to simplify the display of large designs is to change the way arrays are displayed. You can specify that only a maximum number of cells in an array are drawn. You can also specify that only the cells on the sides of arrays are drawn, or that no cells in an array are drawn. Unlike the view limit feature, these display restrictions are not affected by the display scale.

To see the effects of this feature, you must add an array. First zoom out again so that you can see a clear area to add the corner of the array.

1:**ADD → ARRAY → B0STF →** (ROTATE CODE)**none →** (STEP)**Def'lt**

At the prompt for the number of columns type 20, and at the prompt for the number of rows type 20 again. Position the corner of the array with the mouse and click the left mouse button. Now display the entire array with:

1:(VIEW)**all**

Note that only the bounding box of the array is shown. This is because the view limit has become active at this scale. Turn off the view limit feature with:

1:**VIEW →** (LIMIT)**off**

Now the entire array is drawn. Display only the cells on the sides of the array with the options:

3:**ARRAY →** (DRAW MODE)**sides →** (CELL LIMIT)**NoChg**

Even if you zoom in on a corner of the array, the interior cells of the array will not display. If you use this feature, you must use care to remember that not all of the geometry is displayed. The view limit feature is safer in this respect. When you use the view limit instead, by the time you zoom in enough to work on a specific area, all of the geometry is automatically displayed.

Delete this new array now that we are done with learning how to limit the display of arrays.

1:SELECT**(new)**

1:**DELETE**

Display Grids

You may have noticed that as you zoomed in and out with the previous lessons, the grid displayed in the background changed depending on the display scale.

There are 3 display grids available in ICED™. Grid 1 is the finest grid. It is defined with a step value expressed as a real number of user units. (User units are usually interpreted as microns.) Let us change the current values of grid 1 with the menu options:

3:**GRID 1** → (STEP)**0.5000** → (DOTS)**cyan**

The command history line should report the following information:

```
GRID 1 ON COLOR=CYAN DOTS STEP=0.5
```

This tells you that this grid is turned on, that the color of this grid is cyan, the grid points are drawn as dots, and that they are 0.5 microns apart. Zoom in the display until the scale is fine enough to see both white and cyan dots by repeating the menu option:

1:(VIEW)**in %** → **2.0**

The white dots represent grid 2. Report the settings of this grid by **typing** the grid command with the grid number as the only parameter. (Remember that typing a command with no parameters usually reports the current settings of a feature without changing any settings.)

GRID 2

The command history line should now read:

```
GRID 2 ON COLOR=WHITE DOTS STEP=2
```

The step of grid 2 is expressed as an integral number of grid 1 steps. So grid 2 will draw dots every 0.5 x 2 = 1 micron. These dots are drawn on top of the cyan dots.

You may be able to see some dots drawn in magenta. These are dots on grid 3. Report the settings of this grid by typing:

GRID 3

The command history line should now read:

```
GRID 3 ON COLOR=MAGENTA DOTS STEP=5
```

This indicates that grid 3 will draw magenta dots every 5 grid 2 steps, or every 5 microns. Since these dots are very hard to see, let's change them to crosses with the menu options:

3:**GRID 3** → (STEP)**NoChg** → (CROSSES)**magenta**

Now zoom out slightly and you should see the cyan grid disappear since it is now too fine to be useful. Zoom out more and the white grid will disappear also leaving only the magenta crosses every 5 user units.

Interrupting a Redraw

One other method of controlling the display is important only when drawing much more dense layouts than we used here is the ability to interrupt any redraw operation. Any time the view window is taking a long time to redraw (including the initial display of the layout when the editor is opened) you can interrupt the redraw operation with the <Esc> key. The editor will then be instantly ready for commands.

Conclusion

This concludes the tutorial. Close the editor with the following option:

1:**FILE** → **EXIT**

One subject that affects the display that we did not cover in this tutorial is the display of text components. This subject was covered previously a lesson on page 133.

Plotting Layout Data

This tutorial will cover the steps involved in plotting on printers and plotters.

There are two steps to plot a layout on a printer or plotter. The PLOT command in the layout editor creates a file with a .VEC extension. Next the MkPlot utility in a console window translates this file into a print file and sends it to your printing device.

To make the second step of the process largely transparent, the PLOTNOW.CMD command file can be used to create a plot. This command file executes the PLOT command and then spawns the MkPlot utility automatically. It also saves your choices about plotting device options for succeeding plots.

Despite the fact that the PLOTNOW command file is usually easier to use, there are times when you might prefer to use the two step method by explicitly executing the PLOT command and the MkPlot utility. See the table below.

	PLOT command	**PLOTNOW command file**
Menu option	1:**FILE** → **PLOT**	1:**FILE** → (PLOT)**now**
Extra steps	MkPlot must be executed from a console window.	None
Interactive choices saved for next plot	No (Except that you can create your own command file from the journal entry generated from your menu choices.)	Choice of printing device port name is saved as default for next plot.
Series of plot jobs	Caution must be taken to give each plot in the series a unique name.	Plots in series are renamed automatically.
Bitmaps or other plots where you must select a pixel resolution at run time.	The MkPlot utility will prompt you for the resolution in the open console window.	MkPlot will pause waiting for you to supply the resolution. You must explicitly switch to the temporary window to complete the operation.
Large plots	MkPlot runs much faster when executed directly from the console window.	Execution is slower in the background.

Most of the plots you will create below will be created with the PLOTNOW command file. However, this tutorial concludes with the creation of a bitmap graphic (.bmp) file using the PLOT command/MkPlot utility method.

Whichever method you use, the appearance of layers in a plot depends on settings created by the COLOR and LAYER commands and on the pattern file selected for the plot. We will cover the use of these settings to customize what colors and patterns are used in the plot. These settings can be similar to those used for screen display, or quite different to allow plots to be more readable or more standardized.

For all of the lessons below, we will be plotting a copy of the NAND cell supplied with the installation. Clear the contents of the TUTOR directory if desired, although this is not critical. Open a console window with the ICED icon on your desktop. Use the steps below to change to the TUTOR directory and open the layout editor to create a new cell.

CD TUTOR

ICWIN PLOTCELL

Add a copy of the NAND cell with the following menu options:

1:(ADD)**cell → NAND**

Make sure your printing device is turned on and ready to print.

Selecting a Plotting Device

Before you begin plotting there are two pieces of information you must know about your plotter or printer. The name used by ICED™ to identify the type of printer, and the port or network name of the specific device.

A file with the extension .PDF[29] (Plotter Definition File) exists in the Q:\ICWIN\AUXIL directory for each device type supported. ICED™ occasionally adds support for new printing devices, so rather than list the printer models

[29] This file extension has nothing to do with the Adobe Acrobat Reader utility. We were using this file extension for an entirely different purpose before Adobe released their Reader.

supported at the time of printing this document, look at the current list supplied with your installation. This list is stored in the file Q:\ICWIN[30]\DOC\PLOTTERS.TXT.

Browse this file now with your favorite text editor. To use NOTEPAD.EXE from a console window to do this, open a second console window with the ICED icon on your desktop, then type the following at the console prompt:

NOTEPAD DOC\PLOTTERS.TXT

Look for a printer model that matches your printer or plotter. If you have access to a color printer/plotter, select that model name. Note that most printers that support more than one print resolution have a PDF name for each supported resolution.

For example, if you have a Hewlett Packard Color Deskjet Printer available for your use, you can select either the CJET150 or the CJET300 pdf to print at 150dpi (dots per inch) or 300dpi respectively. If the only printer you have access to is a black and white laser printer, use either the pdf name LJET150 or LJET300 in the examples below. Note the pdf name that most closely matches your device.

Note that this file contains instructions on creating your own pdf file. Even if your device is not in the list, you may not need to create a new pdf file for your device. Try using a few of the pdf files in the list first to see if one of those will work for your device.

Now close the file editor window; do not save changes to the file.

The next piece of information you need is the name used by your computer to identify the specific device. If the device is attached to your computer with a parallel cable, the name is most likely **LPT1**, the name of the port used by the cable.

If your device is accessed through a network, you must use the UNC (Universal Naming Convention) name for the device. This name probably looks similar to **\\*share*\name.**

If you already know the name of your printer, you can skip ahead to Plotting in Outline Mode on page 190.

The method used to determine the printer name varies with the operating system. Try the method indicated next to your operating system below. (Most of these methods begin by clicking the button labeled “Start” on your task bar.)

[30]Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. Replace Q: and \ICWIN with the location you defined during the installation.

Windows 95:	Explorer → Network Neighborhood (to determine the server or computer name that begins the unique printer device name) then, Start → Printers → Properties → Sharing
Windows 98:	Start → Settings → Printers: Right click on the printer → Properties → Details
Windows NT:	Start → Settings → Printers: Right click on the printer → Sharing

Plotting in Outline Mode

There are many customizations you can make to a cell file to set layer colors and patterns for plotting. However, let us begin with the easiest way to plot a cell that already has some solid fill patterns defined for the display.

In the case of the NAND cell that is now displayed on your screen, note that the device layers are drawn with solid fill. If this cell was plotted at full scale on a color desk jet printer, the areas of solid color would use a lot of ink. Far more ink than is really required to easily see the boundaries of the shapes.

We will change the plot fill patterns used for individual layers in the next lesson. However there is an easy way to plot any cell in outline mode without having to change the fill patterns assigned to layers. You always assign a separate pattern file for plotting. If this pattern file has no fill patterns assigned, then all shapes are drawn in outline mode. Such a pattern file is supplied with the installation.

Begin the plot process with the menu options:

1:**FILE →** (PLOT)**now**

You will be presented with a menu of printer types. Select the model you chose from the PLOTTERS.TXT file in the last lesson. We will refer to this menu choice from now on as ***PDFMODEL***.

The next menu allows you to select the pattern file used to create the fill patterns on the plot. Select the option **NONE**. This is an empty pattern file used to force all layers to be drawn in outline mode.

The next option is to choose the size of the paper in the plotter/printer. The sizes listed depend on the pdfmodel chosen above. Choose the smallest size provided, size **A** (8.5 x 11) in most cases.

The last menu allows you to avoid overriding the other options available in the PLOT command. Select **none**.

Now the command file prompts you to enter the name of the specific printer where the print file will be sent. Type the printer name you determined in the last lesson. The value you type will be used as the default for future uses of the PLOTNOW command file. (The pdfmodel and printer name are stored in files in the Q:\ICWIN\SETTINGS subdirectory.)

Once you type the <Enter> key, the plot file (Q:\ICWIN\TMP\PLOTCELL.VEC) is created and a temporary console window is created to execute the MkPlot utility. Since this window is automatically minimized, the only evidence of the execution of this utility is the MkPlot item on the taskbar. Once the file is sent to the printing device the temporary window begins a countdown before it closes. This allows you to view any error messages if things go wrong. The PLOTCELL.VEC file is deleted as the last step.

Since you did not override the scaling options, the entire cell is plotted at a scale to fit on your paper.

Note: If your plot came out with solid fill for some layers, get a recent update of the IC Editors software. The appearance of plots plotted with the NONE.STI pattern file was changed in a recent release of the software.

Assigning Plot Patterns

There are two steps to assigning fill patterns to individual layers on plots.

- Assign a pattern number to plot the layer with the PEN parameter of the LAYER command. (This parameter controlled the pen number back when pen plotters were popular.)
- Choose a file that defines a plot pattern for each pattern number. The selection of the pattern file is made during the PLOTNOW command file (or the PLOT command.)

First, let us look at the current settings of the PEN parameters of the layers in our new cell. These settings were made by the startup command file executed when you created the new cell. To view these settings, use the following menu option:

2: (TEMPLA)**screen**

The end of the listing on the screen displays the layer settings. Note that most of the layers are defined with **PEN=***. This setting indicates that the plot pattern number is the same as the screen pattern number indicated after the PAT keyword.

Pattern numbers 0 and 1 are never defined in a pattern file. They have a special purpose:

Pattern number 0	**Draw outline of shape only**
Pattern number 1	**Fill outline of shape with solid color**

For our NAND cell, several layers are defined with pattern 1. For plots, small shapes (e.g. via shapes or contact holes) drawn with solid fill are acceptable, but larger shapes may look better drawn with a sparse fill pattern rather than with solid fill. Large shapes plotted with solid fill patterns may result in so much ink deposited on the paper that the paper wrinkles. We will change the plot fill patterns for the NDIF, PDIF, and POLY layers. These layers all currently have the screen pattern 1 (solid fill) assigned to them.

What pattern numbers should you assign to these layers? That depends on the patterns defined in the pattern file you will use for the plot. The menu to assign a plot pattern does not display a sample of each pattern since you assign the pattern file at plot time and the patterns you will use are not known at this time.

However for this example, you will use the same pattern file for plotting as the one assigned for the screen display, SAMPLE.STI. To see the patterns for each pattern number in this pattern file, we will begin the command you would use to change the screen pattern, but we will not actually change the screen pattern assignment. Use the menu options:

3:**LAYER** → **NDIF** → (SET)**pat'rn**

The first listing displays samples of the first 20 patterns in the SAMPLE.STI file. Note that pattern 14 is a pattern of +'s. This is the pattern we will use for the PDIF layer. However, no pattern of –'s is currently shown. However, once you click the option

Nxt Pg

you see the next 20 pattern numbers. Pattern number 32 is a pattern of –'s. We will use this pattern for the NDIF layer. We will use pattern number 20 for the POLY layer. These patterns are shown in Figure 61.

Cancel this command now by pressing both mouse buttons at the same time.

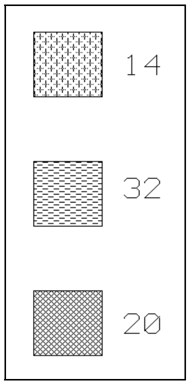


Figure 61

To change the plot patterns for these layers, use the menu options:

3:**LAYER** → **NDIF** → (SET)**pen** → (PEN)**Nxt Pg** → (PEN) **32**

3:**LAYER** → **PDIF** → (SET)**pen** → (PEN) **14**

3:**LAYER** → **POLY** → (SET)**pen** → (PEN) **20**

Since we set up the printer defaults in the last lesson, the process to plot the cell is easier this time. The pdfmodel you selected last time will be indicated at the top of the printer list in a gold color and it will already be selected, so that you only need to quickly press the mouse button to assign it. (If size A is not an option for your device, simply select the first size in the list.)

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**none**

At the prompt, just type the <Enter> key to use the same printer name you used the last time. When the plot is complete look at it carefully.

Depending on your printer and resolution, the overlap of the M1 and M2 layers in the center of the cell may be plotted very densely, making it difficult to see the via shape underneath. Look again at the patterns assigned to the M1 and M2 layers with the menu option:

2: (TEMPLA)**screen**

M1 is assigned pattern 2 and M2 is assigned pattern 3. These patterns are shown in Figure 62. The patterns consist of dense slanting lines slightly offset from each other. On the screen this makes for a very visible display, but on a plot, the overlap area is too dense. We will change the plot pattern for M1 to a less dense slanting pattern and the M2 layer to less dense lines that slant in the opposite direction.

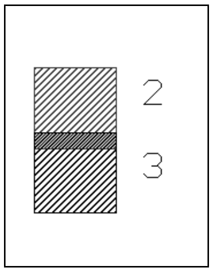


Figure 62

To determine an appropriate pattern number, use the same method you used above. Begin a command to change the screen pattern to display the list of available patterns.

3:**LAYER** → **M1** → (SET)**pat'rn**

After noting that pattern number 7 is appropriate for M1 and pattern 13 is appropriate for M2, cancel the command by pressing both mouse buttons. Now change the plot pattern for these layers with the menu options:

3:**LAYER** → **M1** → (SET)**pen** → (PEN) **7**

3:**LAYER** → **M2** → (SET)**pen** → (PEN) **13**

Now plot the cell again with the options:

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**none**

Press <Enter> and look the plot when it is finished. Now that we have assigned better plot patterns, you could even plot this cell in black and white, and the layout would be easily read.

Overview of Plot Colors

Just as you can refine the plot patterns to be different from the screen patterns, you can refine the plot colors to be different from the screen colors. The startup command file that created this cell used plot color definitions that mimic those on the screen. There are two exceptions to this behavior. The colors white and yellow are handled differently so that these light colors show up on white paper

White	plotted as black
Yellow	plotted as yellow, but a black outline is added at the border of the shape

For the screen display, each of the 16 colors has a palette definition that determines the color seen on the screen. For plots, each color has two definitions. Both definitions are based on dots of the 8 colors available on color inkjet printers shown in Figure 63. The two plot color definitions are as follows:

ONE_DOT Each pixel of the plot is drawn in one of the colors shown in Figure 63.

FOUR_DOTS Each pixel of the plot is drawn as a super-pixel of four dots. Each of the four dots can be different colors from the table in Figure 63.

Black
White
Yellow
Green
Red
Cyan
Blue
Magenta

Figure 63: Printer colors

The four dots definition allows you to define plot colors for color numbers 8:16 that appear quite different from the colors in the table even though those are the only colors available to a color printer. The price of these extra colors is that the apparent resolution of the plot is only half of the resolution used when the one_dot mode colors are used.

For example, if you have a 300dpi color printer, and you use the one_dot colors, you will plot 300 pixels per inch. However, if you use the four_dots colors, you will see only 150 super-pixels per inch.

Let us look at the color definitions in our sample cell. Use the menu option:

2: (TEMPLA)**screen**

Note that the one_dot and four_dots definitions are together in a block below the color palette definitions. Let us look more closely at the definition of the color orange.

```
COLOR ORANGE ONE_DOT=RED FOUR_DOTS=(RED,YELLOW,YELLOW,RED)
```

This color definition results in the shapes on an orange layer being plotted in red when the one_dot definition is used. When the four_dots definition is used instead, one orange plot pixel is drawn as four dots: two yellow dots and two red dots.

You select which color definition is used in a plot with the SUPER-PIXEL MODE menu option. (In a typed PLOT command, the DOTS=1 parameter selects one_dot mode, while DOTS=4 plots in four_dots mode.)

Plotting in FOUR_DOTS Mode

Skip this lesson unless you have access to a color printer/plotter. So far you have been plotting in one_dot mode. This is the default. When you are plotting without solid fill patterns this mode is usually preferable. However if the scale of your plot is very fine, the patterns may be hard to see, and having more unique colors and solid fill patterns would make the plot easier to read.

First, add an array of our NAND cell to the sample cell so that we can plot a denser design.

1:**ADD → ARRAY → NAND →** (ROTATE CODE)**none →** (STEP)**Def'lt**

At the prompts for the number of columns and the number of rows, type

50

3

Now let us reset the layer definitions we changed in the pattern lesson back to solid fill patterns for plotting the NDIF, PDIF, and POLY layers.

3:**LAYER → NDIF →** (SET)**pen →** (PEN) **pen=***

3:**LAYER → PDIF →** (SET)**pen →** (PEN) **pen=***

3:**LAYER → POLY →** (SET)**pen →** (PEN) **pen=***

Now change the color assigned to the PDIF layer. Right now this color is YELLOW. This color is somewhat difficult to see on color plots, and the black border that is added can be confusing. Use the color Orange instead. This color might have been a problem in one_dot mode plots since it plots in the same red color as the red POLY layer. However, in four_dots mode, the color will be easily distinguished from red.

3:**LAYER → PDIF →** (SET)**color →** (COLOR) **10 →** (PAT'RN) **No Chg**

To plot in four_dots mode, you must set one of the options that we have been skipping so far. So under the OPTIONS menu of the PLOTNOW command file, be sure to select **some** this time.

1:**FILE →** (PLOT)**now →** (PLOTTER)***PDFMODEL →***
(PAT'RN)**SAMPLE →** (SIZE)**A →** (OPTIONS)**some →**
(SUPER-PIXEL MODE)**4 DOTS →** (NX NY)**1 1 →**
(TEXT HALOS)**on →** (ALL)**english**

You will have to press <Enter> twice this time; once to verify the printer name, and once to verify that you approve of the calculated scale to fit the layout on the paper.

Once the plot is complete, note the colors. If you looked at the orange areas with a magnifying glass, you would see that these areas are really alternating pixels of yellow and red. Note that the vias drawn with the color gray, really look gray now compared to the one_dot plots you made earlier.

Note also that this plot was rotated automatically. ICED™ automatically rotates plots that plot at a larger scale when rotated, given the size of the design and the size of the paper.

Using the DITHER Option for Large Format HP Plotters

Skip this lesson unless you are using one of the large format plotters available from Hewlett-Packard, specifically the HP500/800/5000 series plotters. These plotters now support plotting with extra colors beyond the 8 standard colors. However, to use these colors you must allow dithering to be applied to the image. Dithering may result in a difficult to read plot.

In dithering mode, the four dots colors are used, but the average of the four colors specified in the FOUR_DOTS parameter is used as the original color of a single pixel of that color.

Your design may plot better with dithering, or not, depending on the colors you use and the dithering algorithms of your particular model. Try one plot with the dithering option turned on to see if this option improves the readability of your plots.

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**some** →
(SUPER-PIXEL MODE)**DITHER** → (NX NY)**1 1** →
(TEXT HALOS)**on** → (ALL)**english**

Plotting a Certain Area

Suppose that you do not want to plot an entire cell, but only a portion of it. This is easily accomplished with the PRESCALE option. There are a few extra steps involved in selecting the area and the scale of the plot. The general process is as follows.

- You to use the mouse to indicate the approximate corners of the area to be plotted.
- The program then calculates the largest scale factor that will allow the area to fit on the indicated paper size.
- You can then override the calculated scale factor to use a more exact number.
- The program then recalculates the largest rectangle that will fit on the paper and you can choose to reposition the plot window rectangle over your layout to select the exact area to be plotted.

This is easier to do than to explain, so let's begin.

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**some** →
(SUPER-PIXEL MODE)**1 DOT** → (NX NY)**1 1** →
(TEXT HALOS)**on** → (PRESCALE)**english**

The **english** and **metric** options under each scale option refer to the units of the scale factor for the plot. The english option will use a scale factor expressed in mils/user unit. If you prefer to see the scale factor expressed in microns/user unit, use the metric option instead.

Once you have made the menu choices above, you press <Enter> to use the same printer name you have used in previous plots. You are then prompted to select the approximate corners of the plot with the mouse. Click the left mouse button at one corner, then again for the other corner.

You are then prompted to override the plot scale factor. If you simply press <Enter>, the scale factor shown in square brackets will be used. For example if the prompt reads:

```
Enter plot scale in mils per user unit [37.244]:
```

then 37.244 millions of an inch will be used to plot one user unit. Unless the exact scale of the plot is important, just press <Enter>.

The next prompt asks if you want to reposition the plot window. If you did override the scale factor, you may want to respond with a <Y> key to this prompt. This allows you to use the mouse to carefully position the exact plot window rectangle which has been recalculated for the new scale factor. However, if you use the calculated scale factor, or if the exact boundary of the plot is not critical, respond with a <N> key to continue.

This concludes the plot process and the plot is sent to your printer.

If you want to plot with an exact scale factor, you use the options under (SCALE) rather than those under (PRESCALE). This process is largely the same as PRESCALE, except that the first steps of approximating plot window and the scale are omitted. You are prompted for the scale, and then you place the exact plot window rectangle with the mouse.

Plotting at a Larger Scale over Several Pages

Suppose that you need to plot a large design at a relatively fine scale. If you are working with one of the large format printers, there is no problem. In this case, you can skip this lesson. However, if you are using a printer that uses only 8.5x11 paper then you need to use the NX and NY options to plot over multiple pages.

To plot over multiple pages, you need to select the number of pages in the x and y directions. You can usually guess at the best ratio of pages in the x-direction to the number of pages in the y-direction. You can also use the 2:RULER option to measure the entire design to calculate the ratio. However, let us use the plot window display of the PRESCALE plot option to perform this selection visually. We will guess that 4 pages in the x-direction and 1 page in the y-direction will plot our design at a fine enough scale without a lot of wasted whitespace on the paper. Then we will see the actual plot window on the screen before sending the plot job to the printer to test our guess.

The NX and NY options control how many pages are used to plot the design. We have been using the default values of NX=1 and NY=1. These defaults can be overridden with the menus when you select **some, most,** or **all** under the OPTIONS menu of the PLOTNOW command file.

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**some** →
(SUPER-PIXEL MODE)**1 DOT** → (NX NY)**4 1** →
(TEXT HALOS)**on** → (PRESCALE)**english**

After you press <Enter> to use the saved printer name, you will need to digitize a box that covers the entire design. If you cannot draw the box around the entire design easily, you can use the nested view menu (by pressing the <Esc> key) to change the view window to display the entire design (option NVM:**ALL**) without canceling the current plot command.

You approve the calculated scale factor by pressing <Enter>, then be sure to reply with a <Y> to the prompt allowing you to reposition the plot window. Note that the plot window rectangle is quite a bit taller than needed to cover the entire design. This will lead to wasted white paper. A value of 5 for NX would lead to a better plot. Cancel the current command by pressing both mouse buttons.

Now begin the process again. This time you will need to supply the NX and NY parameters with the keyboard since 5 is not offered on the menu. You will be prompted to type in the parameter values after completing the menu choices.

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**some** →
(SUPER-PIXEL MODE)**1 DOT** → (NX NY)**KeyBrd** →
(TEXT HALOS)**on** → (PRESCALE)**english**

This time the plot window is closer to the shape of the array. Less white space will be wasted, and you may note that the plot scale is larger than the 4x1 plot you canceled above.

Plotting Only Certain Layers

Suppose that you need to plot only the wiring layers of a chip. Perhaps you are searching for a short, and you want to take a careful look at only the layers where the short is most likely to be found.

This is easily accomplished with the LAYERS option. This option is available only when you select **most** under the (OPTIONS) heading.

1:**FILE** → (PLOT)**now** → (PLOTTER)***PDFMODEL*** →
(PAT'RN)**SAMPLE** → (SIZE)**A** → (OPTIONS)**most** →
(SUPER-PIXEL MODE)**1 DOT** → (LAYERS)**layers**

At this point you are presented with a menu to allow you to choose the layers to be plotted. (All of your previous plots were plotted with the default option of all layers.) Click on each layer to be plotted, and then click **Return**.

POLY → M1 → M2 → CONT → VIA →Return

Note that each layer name turns gold after it is selected. Now you can continue the plot process with the following options:

(NX NY)**1 1** → (TEXT HALOS)**on** → (ROTATE)**best** → (ALL)**english**

Creating a Bitmap Graphic File

If you need to add plots of your design to other documents, you can create a bitmap graphic file of your plot rather than sending the plot to the printer. The graphic file will be saved in a file with a .bmp extension. You can add this file to any document created with desktop publishing software as an imported graphic. You can also view the file outside of the ICED™ layout editor and even send the graphic file to others via e-mail.

The PLOTNOW command file is not intended for bitmap export. This command file expects you to supply the name of a valid printing device, which you do not need for bitmap export. Also, you will need to supply the resolution of the bitmap file to the MkPlot utility. If you use the PLOTNOW command file, this utility executes in the background, and you may wait for a long time before you realize that the utility is paused, waiting for input from you.

Instead, you will use the PLOT command and explicitly execute MkPlot from the console window. This method can be used to create any plot. Since the MkPlot utility executes more quickly when executed with this method, you may want to use this method any time you have a large plot to print.

To begin the bitmap export process, use the menu option PLOT **above** the (PLOT)now option you have been using so far. Select the BMPCOLOR option rather than the *PDFMODEL* printer model name you have been using,

1:**FILE → PLOT →** (PLOTTER)**BMPCOLOR →**
(PAT'RN)**SAMPLE →** (SUPER-PIXEL MODE)**1 DOT →**
(SIZE)**A →** (OPTIONS)**none**

The command completes at this point with no further interaction. However, the file created is still in .VEC format. Note that the history line tells you the name and location of this file; Q:\ICWIN\TUTOR\PLOTCELL.VEC. You must execute the MkPlot utility to translate it into a usable bitmap graphic. Open another console window with the ICED icon and type the following:

CD TUTOR

MKPLOT PLOTCELL

The utility prompts you for the resolution of the file. You can supply any non-zero value. There is no benefit to supplying a value higher than the resolution of your printer, unless you will be scaling the bitmap to a larger area before printing. You will create a larger bitmap file, but the extra pixels will be lost as the file is translated to your printer. If you will be printing the document that will contain the bitmap file on a 300dpi printer, use that value as the resolution of your bitmap file. Once you supply the resolution, the utility translates the file and completes.

The bitmap graphic has been created with the name PLOTCELL.BMP. You can view this file with an appropriate utility. If you have access to the Microsoft PAINT utility (MSPAINT.EXE), you can view it with that utility. If you have access to Microsoft WORD, you can import it into a document with the INSERT → PICTURE menu option. If you have access to Netscape, you can view it by creating a blank page, then clicking the IMAGE button. You can even assign this file as the background graphic on your desktop with START → SETTINGS → CONTROL PANEL → DISPLAY.

Conclusion

The process to create a plot can be cumbersome since there are so many options. Once you have selected all of the menu options to create a plot, a PLOT command is created in the journal file. If you need to repeat the exact same plot command, you can copy this PLOT command from the journal file into a command file. Then you can execute the saved PLOT command by simply executing the command file. You will not need to make all of the menu selections each time. We describe this general process in the tutorial on command files on page 226.

If you want to learn about plot options we have not covered in this tutorial, see the description of the **PLOT** command in the Layout Editor Reference Manual. You can learn about filtering shapes by size and various text plotting options among other subjects.

If you want to create custom pattern files, read about the MkSti utility in the Layout Editor Reference Manual. You can use different pattern files for screen display and for plotting so that the same pattern numbers are optimized for each different device. For example, a pattern file for a low resolution color printer might be quite different from one optimized for a high resolution black and white laser printer.

If you want to learn more about plot colors, read the descriptions of both the COLOR and PLOT commands carefully.

If you are plotting a cell file created with an old startup command file, the ONE_DOT and FOUR_DOTS parameters of the COLOR command may not be defined for your cells. In this case, you may want to look at the Q:\ICWIN\AUXIL\COLORS.CMD command file. Copy and modify this file as required to define your plot colors, then execute it in every cell you need to plot.

This concludes the tutorial. Close the editor with the following option:

1:**FILE → LEAVE**

Importing and Exporting Data with GDSII-Stream Files

This tutorial focuses on the creation of ICED™ cell files from data stored in Calma GDSII-Stream format files and the export of stream files from ICED™ data.

Stream files are imported with the **UnStream.EXE** utility supplied with the ICED™ installation. This utility is executed as a two-step process. The first pass reads the stream file and creates an editable ASCII **alias file** that allows you to adjust the import process by overriding layer assignments and structure name-cell name correspondences. The second phase of the import process uses this alias file and your responses to several interview questions to create the cell files.

The first lesson demonstrates the simplest method to import a stream file. This method is particularly useful when you have no access to information about how the design in the stream file is organized. You will use defaults for all of the UnStream utility's parameters to get cells files created with no user interaction. You will then use the layout editor to interpret the data created.

However, the UnStream utility has several parameters that allow you to create a more exact translation (or even transformation) of the layout data contained in the stream file. The more information you have about the design organization, the more you can control the translation. If you have a description of the layers used, you can use this information to define a layer mapping file and a startup command file to initialize the layer environment of all cells. If you know that the stream file contains non-standard structure names, you can control how these names are translated to ICED™ cell names in the alias file. If you know that the design uses other non-standard features, you can change the default behavior of the utility by overriding parameters in the interview questions presented in the second pass of the utility.

Most of the interview questions asked by the UnStream utility were added to take care of rarely encountered, non-standard stream file formats. Most stream files can be imported using defaults for all of these options.

We will also cover examples of generating stream format files from ICED™ cells with the STREAM command. One lesson covers stream file export suitable for submission to a foundry. This process includes defining stream export information for every layer with the LAYER command prior to the export.

Simple Import of an Unfamiliar Stream File

Let us assume that you need to import an unfamiliar CMOS design stored in stream file format. Since you do not know anything about how the layers are defined or if there are any non-standard features in the file, you will use defaults for all of the parameters that UnStream uses to translate the data.

The UnStream utility is run in a console window, outside of the layout editor. Open an appropriate console window now with the ICED icon on your desktop.

The UnStream utility always creates cell files in the current directory. If cell files already exist in the current directory, they will be overwritten without warning. So, you should always use an empty directory when performing an UnStream operation. Make this directory the current directory and then copy the stream file to this new directory. The DOS commands below will perform these steps when typed in the console window.

CD TUTOR

MD STREAM

CD STREAM

COPY \ICWIN[31]\SAMPLES\74181\B2STF.SF

Now execute the UnStream utility twice with the /d option that tells the utility to use defaults for the answers to all of the interview questions. Even though you will not edit the alias file created by the first pass of the utility, you must still execute the utility twice to create the cell files. You probably have DOSKEY running in this console window, so you can press the <↑> key to repeat the command without retyping the command line.

UNSTREAM B2STF /D

UNSTREAM B2STF /D

The highest level cell is listed first in the list of cells created by the utility. (For designs that have so many cells that this line scrolls off of the display, you can look at the end of the alias file (*stream_file*.ALI) for the name of the highest level cell, or use the ICTOP utility described on page 153.) Note that the name of the highest level cell in our lesson is "B2STF". Open the layout editor to look at the design with the command:

ICWIN B2STF

[31] Remember that \ICWIN is used to represent the directory where you have installed the ICED™ software. Replace this directory name with the appropriate name.

We have a problem at this point. Since we used the batch file supplied with the program to open this cell file, the default startup command file was used to define the environment of this new cell file. This startup command file includes layer definitions that have nothing to do with the layers in this stream file. Note that most of the layers are drawn in yellow outlines, but at least one of the layers is drawn in green with solid fill. If you use the 2:(TEMPLA)screen option to display the layer information, you would see layer definitions that were not included in the stream file. Since these definitions are completely unrelated to the information in the stream file, they add only confusion and a potential cause of errors.

To get rid of the inappropriate layer definitions created by this startup command file we have two options: reopen the cell without a startup command file or remove the layer definitions with the INITIALIZE command. The second method is more useful since the other settings in the sample startup command file (e.g. color, grid, and key definitions) will still be defined in the new cell. To remove all layer definitions, type the following command on the command line:

INIT LAY

Now there are no unique layer definitions in the environment of this new cell. Note that all layers are drawn in yellow.

Defining Layers in an Unfamiliar Design

We will use the data in the design itself to deduce the appropriate layer definitions. From this information, we will create a startup command file with appropriate layer definitions. You can then create a new batch file to open cells using the new startup command file to create the correct environment in all cells created using this stream file and other stream files from the same source.

It is a pretty safe guess that the long horizontal wire at the top is meant to be on the second metal layer, so let us find out what layer number this shape is on.

2:(SHOW)**screen**

Place the cursor on this wire where no other components overlap. Once you click the left mouse button, the display will report that this component is on layer number 51. Type <Enter> or press both mouse buttons to return to the edit window.

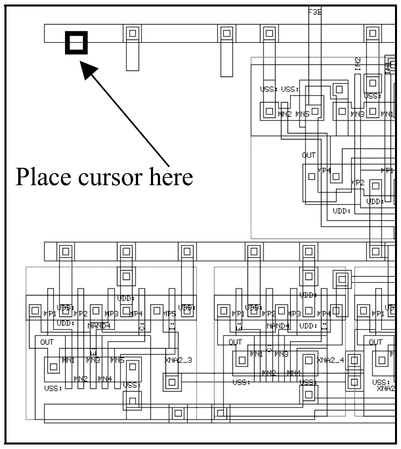


Figure 64: Left half of B2STF cell

Let us apply a suitable layer name, color, and pattern for this layer by typing:

LAYER 51 NAME=M2 WHITE PAT=3

The structures hanging off from this wire are obviously vias and first level metal wires. Determine the layer number of one of the vertical metal wires that cross the M2 wire on the left with:

2:(SHOW)**screen**

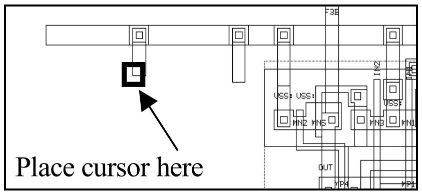


Figure 65: Selecting M1 wire

The layer number is reported to be 49. Define settings for this layer by typing:

LAYER 49 NAME=M1 CYAN PAT=2

The via shape is nested inside of a subcell. We can display the layer number of a shape in a subcell with 2:(SHOW)@deep. First zoom the display in around the area where one of the short M1 wires crosses the M2 wire. Use the menu options:

1:(VIEW)**box**

2:(SHOW)**@deep**

Place the cursor over the via shape and press the left mouse button. The layer number of the shape is reported as 50. Press both mouse buttons to return to the display window and then define this layer with the command:

LAYER 50 NAME=VIA GRAY PAT=1

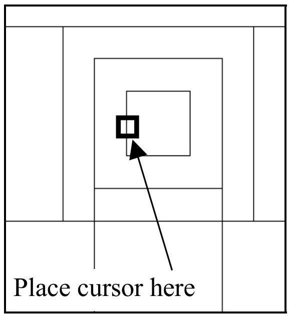


Figure 66: Selecting nested via

You can continue to use these commands to determine the layer numbers of shapes in the circuit. Use 2:(SHOW)screen for shapes in the current cell and 2:(SHOW)@deep for shapes that are nested in other cells. We will assume that you have done this and just tell you the other appropriate layer definitions to type.

LAYER 2 NAME=NDIF GREEN PAT=20

LAYER 3 NAME=PDIF BLUE PAT=20

LAYER 41 NAME=PWEL MAGENTA PAT=0

LAYER 46 NAME=POLY RED PAT=1

LAYER 47 NAME=CPLY WHITE PAT=1

LAYER 48 NAME=CACT WHITE PAT=0

You can see that this design uses as-drawn layers (NDIF and PDIF) rather than the foundry layers (ACTIVE, NSEL, PSEL) since two different active layers are used to form devices. You can tell which is which from the power busses that connect to the circuits (NDIF devices connect to VSS; PDIF devices connect to VDD.)

Now view the entire cell with the menu entry:

1:(VIEW)**all**

Now that our layer definitions are made in this cell, you can export the layer definitions to a command file with the menu option:

2:(TEMPLA)**file**

At the prompt, type the file name as:

TUTRCMOS.CMD

Close the editor now and save this cell with the option:

1:**FILE → EXIT**

You can now use the file created above as your startup command file for other cells created from this stream file, or other stream files from the same source. This will automatically add the layer definitions you just created to any cell you open. You can make a copy of the sample project batch file (ICWIN.BAT) then edit the command line to use this new startup command file. (See the lesson on page 63 to learn how to edit project batch files.)

One trick you might remember when performing this type of process on a larger, denser design is to look at a simple cell to determine the layer definitions. Inverter cells are ideal for this purpose. The circuitry is usually easy to follow. Most designs have at least one cell that you can tell is an inverter cell from the name of the cell.

Determining the Resolution of an Unfamiliar Design

You may want to make other changes to a new startup command file before using it for design work. For example, the default grid resolution in the sample startup file is 0.5. This is sufficient for the design we just looked at in this lesson. However, many newer designs use a much finer resolution than this.

To see if your real imported design needs a finer grid resolution, edit a dense cell with many devices. Select all of the components and then use 2:(SHOW)screen to look at the coordinates of these shapes. Note the coordinates that have numbers after the decimal points. If all coordinates end in .5 or .0, then the default resolution is sufficient. If some coordinates end in .1 (or other single digits past the decimal), change the grid resolution and grid snap to a .1 step. If coordinates end in .11 (or other double digits past the decimal), change the grid resolution and grid snap to a .01 step.

The resolution grid is changed with the RESOLUTION command. The snap grid is set with the SNAP command. Make these changes to your new startup command file before you attempt to edit any cells. Alternately, you can make these changes to the cell before exporting the new startup command file as we did in the last lesson.

Two-Step Import Using Layer Information

The B2STF design we imported earlier has components on two layers that we did not redefine. There are text components on layers 60 and 62. Let us suppose that we would prefer that all these text components were created on layer 33 instead. We could use SWAP commands to change the layers of these components, but we would have to perform the command in every cell.

It is much easier to change the layer for these components automatically in all imported cells by changing the layer correspondence in the alias file during the import process.

Remember that the UnStream utility must be executed twice to actually create cells from a stream file. You can edit the file created by the first pass before executing the second pass. You can change layer correspondences and/or structure name-cell name correspondences in the alias file before cells are created.

To see how this works, let us delete all of the cells and the alias file we just created in the STREAM subdirectory. (Make sure that the TUTOR/STREAM directory is still the current directory before typing the following DEL commands.) In the console window type:

```
DEL *.CEL
DEL *.ALI
```

Now begin the stream file import process again with:

UNSTREAM B2STF /D

Now edit the alias file created by this step with:

NOTEPAD B2STF.ALI

This file contains copious comments on how to use the file. You can read these now if you desire. (If you edit an alias file for a real design, read everything thoroughly.)

Scroll down to the bottom of the file where the lines that begin with T and D are located. These lines define the stream layer number – iced layer number correspondences. Lines that begin with 'T' define stream layers that contain only text components. Lines with 'D' represent layers with component data. We will change only the last two 'T' lines.

The format of the 'T' lines is:

T ***stream_text_layer_number text_type iced_layer_number***

All text types are 0 in the stream file, or lines for other text types would have been created in the alias file. We want to change the *iced_layer_number* of stream layers 60 and 62 to ICED™ layer 33.

Old:	T: 60 0 60
New:	T: 60 0 **33**
Old:	T: 62 0 62
New:	T: 62 0 **33**

Now stream text components on both layers 60 and 62 will create ICED™ text components on layer 33. **Save the file** and close the Notepad editor.

Now execute the second pass of the utility. (You can use the <↑> key twice to retrieve the command.)

UNSTREAM B2STF /D

Now open the B2STF cell in the layout editor. If you created a new project batch file with the new startup command file we created, use that batch file to open the editor. If not, just use the ICWIN.BAT file.

ICWIN B2STF

If you used ICWIN.BAT, execute the corrected startup command file by typing:

@TUTRCMOS

Zoom in to see the text components. Zoom in on the M1 wire sticking up above the top row. This wire has a text component on it. Then display the layer of this component with:

1:(VIEW)**box**

2:(SHOW)**screen**

You can see that the text component is now on layer 33. Now close the editor:

1:**FILE → EXIT**

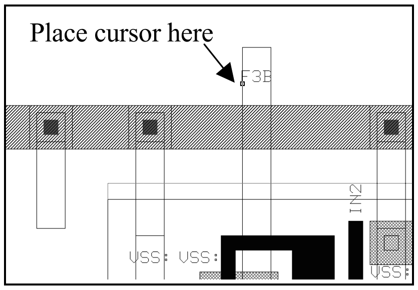


Figure 67: Selecting text for SHOW

Using a Layer File in the First Pass of UnStream

If you would be unstreaming several files from the same source, you could use the layer lines from the alias file you edited above as a layer correspondence file in the first pass of the utility. Then you would achieve the same result without having to edit the alias file each time.

Edit the alias file again.

NOTEPAD B2STF.ALI

Delete all lines except for the T and D lines. (You can add comment lines as long as the first character on the line is '!'.) **Save the file as TUTRCMOS.LAY** and close Notepad.

Now delete the files created by the last execution of the utility and re-execute it **without the /d option** that uses defaults for all translation parameters.

DEL *.CEL

DEL *.ALI

UNSTREAM B2STF

Use the default value (indicated in square brackets) for most of the parameters by pressing <Enter> after each prompt. However, when prompted for the layer file name, type the string in bold type below.

```
Enter optional layer file name [NONE]:
```

TUTRCMOS

For the second pass, add the /d option back to the command line.

```
UNSTREAM B2STF /D
```

Look at the cell if you want to. It is identical to the design created by our previous use of UNSTREAM. The text components that were on stream layers 60 and 62 were created on ICED™ layer 33.

The STREAM Command

Stream files are created from the data in ICED™ cells with the STREAM command in the layout editor. This command has two options for how ICED™ layers are mapped to stream layer numbers:

MAP_LAYERS	Use parameters in the ICED™ layer definition to determine stream layer numbers, data types, and text types.
ARCHIVE	Ignore stream layer number definitions in the cell and map each ICED™ layer to a stream layer with the same number. No data types or text types will be created.

The MAP_LAYERS option should be used to create stream files you will use to export your design to a foundry. The ARCHIVE option is intended for backups of your design or for compact exchange of data between ICED™ users. You can also use this option in combination with the UnStream utility to strip the environment database from all cells in a design.

The options are used for such different purposes, they are listed separately on the menus. The STREAM MAP_LAYERS command is executed with the 1:FILE → STREAM option, while the STREAM ARCHIVE command is executed with the 1:FILE → ARCHIVE option.

Both types of Stream export are intended for export of the root cell. All subcells of the root cell are included by default. If you want to restrict the stream file to only the root cell, use the ROOT option. If you want to include only subcells in the same directory as the root cell, use the NO_LIBS option instead.

You cannot export a stream file for a subcell that you opened with the T_EDIT or P_EDIT commands. However, if you open a subcell with the EDIT command, you can export a stream file for that subcell.

Creation of a Stream File Suitable for a Foundry

The most important aspect of a stream file export process with layer mapping is to define the stream layer number (and data type and text type) for each ICED™ layer to be exported. This is done with LAYER commands prior to the export of the stream file.

If your project is set up correctly, you should assign the stream layer numbers in the startup command file used for all designs of a given technology. In any case, it is critical to assign these numbers carefully in the cell from which you are doing the export. If you do not assign these numbers in your startup command file, it is best to create a separate command file that assigns these numbers. You then execute this command file in a cell before you issue the STREAM command. We will use this method for this lesson.

In either case, the **stream number assignments are not saved with the cell file**. They are deliberately forgotten when you close a cell. This is because of the risk of the stream number assignments varying from one cell to another in the same design, (not an uncommon occurrence if the startup command file changes during the course of a design). **You must redefine the stream number assignments in the current editor session to export a stream file with the MAP_LAYERS option.** If you forget to redefine these stream numbers, the STREAM command will fail with the message “No layers assigned Stream Format layer numbers.” Only layers with stream layer numbers defined for them will be included in the stream file. Other layers are ignored without warning.

Create a new directory and make this directory the current directory, then copy all the cell files in Q:\ICWIN\SAMPLES to this empty directory with the command:

MD STR_OUT
CD STR_OUT
COPY Q:\ICWIN[32]\SAMPLES*.CEL

Open the NAND cell file with the command:

ICWIN NAND

Look at the existing layer definitions in this cell file with the menu option:

2:(TEMPLA)**screen**

[32]Remember that Q: and \ICWIN are used to represent the drive and directory where you have installed the ICED™ software. Replace Q: and \ICWIN with the location you defined during the installation.

You can see that all of the named layers have the parameter “NOSTREAM” assigned to them. This means that none of these layers can currently be exported to a mapped stream file. You must assign valid stream numbers to each layer to be exported.

We will be performing these stream number assignments in a command file. Create a file that is easily edited to perform these assignments with the menu option:

2:(TEMPLA)**file**

At the prompt, type the following file name:

STREAMNUM.CMD

Now edit this new file with your favorite text editor. You can open a new text editor window right from the layout editor command line by typing the command:

SPAWN NOTEPAD STREAMNUM.CMD

Delete all lines except for the LAYER command lines. Also delete the lines with the definitions of LAYER * and LAYER 0. Delete all of the parameters except for the layer names and the NOSTREAM keywords. The file should now look like Figure 68.

```
LAYER 1  NAME=NWEL  NO_STREAM
LAYER 2  NAME=NDIF  NO_STREAM
LAYER 3  NAME=PDIF  NO_STREAM
LAYER 4  NAME=PSEL  NO_STREAM
LAYER 5  NAME=POLY  NO_STREAM
LAYER 6  NAME=M1    NO_STREAM
LAYER 7  NAME=M2    NO_STREAM
LAYER 8  NAME=CONT  NO_STREAM
LAYER 9  NAME=VIA   NO_STREAM
```

Figure 68: STREAMNUM.CMD

We will be replacing each “NOSTREAM” parameter with a valid stream number definition. The correct syntax of each stream number definition is as follows. (The parameters in square brackets are optional.)

STREAM=*stream_layer_no* [,*stream_data_type* [,*stream_text_type*]]

All components on a given ICED™ layer will be mapped to the same stream number. However, maskable components will be given the indicated data type number while text components on the same layer will be given the indicated text type number. Both the data type and text type default to 0 when they are not supplied in the LAYER command.

More than one ICED™ layer can be mapped to the same stream number. This allows you to create layers with the same stream layer number, but different data types if you require this. However components with different data types must be on different ICED™ layers. (Remember that selected components on a given layer in the current cell can be swapped to a new ICED™ layer with the SWAP command.)

Let us assume that all data types should be 0, but all text types should be 5. Change the lines in the file to match the lines shown in Figure 69.

Save the file, close the text editor, and execute the file in the layout editor with the menu option:

3:**@%.cmd → STREAMNUM**

```
LAYER 1  NAME=NWEL  STREAM=1,0,5
LAYER 2  NAME=NDIF  STREAM=2,0,5
LAYER 3  NAME=PDIF  STREAM=3,0,5
LAYER 4  NAME=PSEL  STREAM=4,0,5
LAYER 5  NAME=POLY  STREAM=5,0,5
LAYER 6  NAME=M1    STREAM=6,0,5
LAYER 7  NAME=M2    STREAM=7,0,5
LAYER 8  NAME=CONT  STREAM=8,0,5
LAYER 9  NAME=VIA   STREAM=9,0,5
```

Figure 69: Corrected stream definitions

(If we did have stream number assignments in our startup command file, we would not have needed to create a separate command file. We could just have re-executed the current startup command file in this cell with the 3:@START menu option.)

Now you can export the design in the cell as a stream file with the STREAM command. There are several optional parameters in this command. (See Figure 70.) We will not be overriding any of the defaults for these options in this lesson. However, if you need to change any of these values choose the "(OPTIONS)all" menu option when exporting your stream file.

1:**FILE → STREAM →** (OPTIONS)**none**

The STREAM command generated from your menu choices is shown on the command history line and stored in the log file. You could add this command to the end of the STREAMNUM.CMD command file. In this case, all you need to do to export a mapped stream file is to execute the command file.

Creation of an Archive Format Stream File

If you are not creating the stream file to export mask data to a foundry, you probably do not need to explicitly map ICED™ layers to specific stream layer numbers, data types, and text types. If your stream file is only a compact backup of your design for archival purposes, or for sharing with other ICED™ users, you can use the ARCHIVE option of the STREAM command. In this case, the export process is simpler. You do not need to define the stream numbers in the current session. Indeed, even if stream numbers are assigned in the current cell, they are ignored.

Menu heading	Option	Purpose	Default
DIALECT	Calma Lower Case Names Computer Vision	GDSII:CALMA format with upper case structure names CALMA format with lower case structure names Computer Vision format	Calma
FONT NAME	def'lt KeyBrd	Use GDSII:CALMAFONT.TX Type in the font name	def'lt
FONT HEIGHT	1.0 2.0 etc	Select height of font in microns	1.0
SAMPLE	none KeyBrd	When you need the stream units to be different from a 0.001 micron database unit and a 1.0 micron user unit, supply the name of a sample stream file with the desired unit definitions.	none: units are defined as 0.001 micron database unit a 1.0 micron user unit
WIRE ENDS	preserve force-flush	Use wire types defined in cell Force all wires to flush-end definitions	preserve
OUTPUT	Root NoLibs Full	Export only the root cell, no subcells Export only the root cell and subcells stored in the same cell library Export root cell and all subcells	Full
SCALE	0.1 1 25.4	Size of the user unit in microns	1

Figure 70: Optional parameters in STREAM command

Components on each ICED™ layer will be exported to a stream layer with the same layer number as the ICED™ layer. All components will have 0 assigned as their data type or text type. Components on all ICED™ layers will be exported. Defaults for all of the stream export options listed in Figure 70 will be used.

To create this type of stream file, use the menu option:

1:**FILE → ARCHIVE**

Now close the layout editor.

1:**FILE → EXIT**

Changing the Scale of a Design with UnStream

The UnStream utility used to create ICED™ cells from a stream file can change the scale of the design as it is imported. For example, the design we used in the first lesson has a minimum device dimension of .5 microns. Suppose that you want to scale this design down to a minimum dimension of .3 microns.

Editing every device and then shifting shapes and cells to take advantage of the extra space would be very time consuming. Instead, you can create a stream file of the design, then use the scaling feature of UnStream to scale the design as it is imported. All coordinates are scaled uniformly as the new cells are created.

We will perform this operation on a single subcell for simplicity. However, you can scale an entire design at once if you desire. We will create the stream file from an existing cell file, but this step is not required if you already have the design in stream file format.

Open the cell XOR_1 in the directory we used in the first lesson with the commands:

```
CD ..        ! Current directory should now be \ICWIN\TUTOR\STREAM
ICWIN XOR_1
```

Now use the ARCHIVE option of the STREAM command to create a stream file. This will use the ICED™ layer numbers in the stream file, so that that when we import this stream file back into an ICED™ cell, the layer numbers will remain unaltered. There is no need to redefine the stream numbers with extra LAYER commands.

1:**FILE → ARCHIVE**

This will create the file “XOR_1.SF”. You can now exit the cell with:

1:**FILE → QUIT**

Before we begin the import process, we need to create a new directory for our modified cells. If you executed the UnStream utility in this same directory, **the existing cell file would be overwritten without warning.** Copy the stream file you just created to this new directory. The DOS commands to accomplish this are shown below:

```
MD XORSCALE
CD XORSCALE
COPY ..\XOR_1.SF
```

Remember that importing a stream file is a two step process. We will not need to modify the alias file in any way between the two steps. However, do not add the /D option to the second pass of the utility since we need to override the scaling defaults.

```
UNSTREAM XOR_1 /D
UNSTREAM XOR_1
```

Since the /D option is missing from the second pass of the utility, you will be presented with parameter options to tailor the import process. We want to use the defaults for most of these parameters. When you want to use the default value shown in square brackets, just press <Enter> at the prompt. This is what you will do for the first two parameters.

```
Enter ICED user unit in microns [1.0]:
```

<Enter>

```
Enter divisions per unit [1000]:
```

<Enter>

For the next several prompts, we need to override the defaults to define the scaling parameters.

```
Are you using UNSTREAM to scale this design or snap it to
grid (Y or [N])?
```

<Y>

```
Enter the original feature size [1.0]:
```

.5

```
Enter the final feature size [1.0]:
```

.3

The next prompt defines the new grid resolution for the design. New coordinates will be snapped to the new grid. The new grid resolution must be finer than the original grid resolution of .5 microns or most of the scaled coordinates will be off-grid. For this exercise, we will use a grid resolution fine enough so that the scaled coordinates will all be created on grid without rounding. (We will experiment with a courser grid and coordinate rounding after we complete this simpler example.)

```
Enter snap grid size in microns [0.001]:
```

.01

The next prompt defines a maximum rounding error for each coordinate. Since we have chosen such a fine resolution grid, all coordinates can be located on this grid with no rounding, so we can use the default of 0.

```
Enter maximum allowable coordinate rounding error in
microns [0.0]:
```

<Enter>

Use the defaults for all of the remaining options by just pressing <Enter> until the utility reports that the conversion is complete.

The scaled version of the XOR_1 cell is now created. It's environment (layer definitions, editor settings, etc.) has been stripped by this process. So when you open the cell for editing, the startup command file will be executed. When you perform this process on a real design, it is important to have an appropriate startup command file in place before you open (and then save) any new cell files. Be sure to modify the resolution grids (set with the RESOLUTION and SNAP commands in the startup command file) to appropriate values for the scaled design.

You can look at this modified cell now if you desire.

What if you need the new design to have all coordinates on a .05 grid rather than the finer .01 grid we used above? In this case, snapping some coordinates to this courser grid will require rounding the coordinates. For example, one of the boxes in the cell will have the new corner coordinates (6.6, 5.28) (6.9, 5.58). If you want this box on a .05 micron grid, it would have to be rounded to (6.6, 5.3) (6.9, 5.6). In this case, the rounding has not changed the size of the box. Both coordinates have shifted up so that the box is still .3 microns high. However, some components may be altered in a way that makes the components wider or narrower than they would have been if their coordinates were not snapped to the course grid.

Another, more significant, distortion can occur in nested cells. Suppose that a deeply nested subcell has a component with a coordinate shifted slightly to the right to be on grid. This subcell is a component in a higher level cell. Suppose that the location of the subcell in the higher level cell is also shifted slightly to the right. If each cell in the hierarchy is shifted slightly to the right, then each tiny displacement adds to the next, and the final location of the component in the main cell could be significantly shifted relative to components in other cells. It is rare for this type of distortion to add up to much more than a single step in the resolution grid, but larger distortions can occur (and they have.)

If distortions like these do not concern you, you can have the UnStream utility snap the coordinates to a course grid with a maximum rounding error as large as the grid resolution. (E.G. use a maximum rounding error of .05 when defining a grid resolution of .05.) In this case, the coordinates will be rounded to the snap grid with no warning messages.

If you prefer that each coordinate that does not wind up on (or very near to) a point on the snap grid results in an error, use the default maximum rounding error of 0 (or a very small non-zero maximum rounding error). In this case, a component with a scaled coordinate not on (or very near to) the snap grid will be reported in the error log file (*sf_file_name*.err) and the component will not be created in the cell file. Unfortunately, the error log reports the missing component using it's original coordinates in the stream file, so you must look at the original cell (before scaling) to make sense of the coordinates in the error log.

If the error log does not help you identify the missing components, another option is to perform the UnStream translation twice in different directories. Perform the translation once with the desired grid resolution and a zero (or very small) rounding error.

Then in a new directory, perform the translation again so that no components are lost. You need to create unique cell names for this second version so that the cell names do not collide. This is easily accomplished by assigning a cell name suffix during the first pass of the UnStream utility. (Do not use the /D option on the command line and you will be prompted for a cell name suffix.) Edit the alias file before the second pass of this second translation so that all layers are translated to some dummy layer number. During the second pass, either use a very fine grid, or a large maximum rounding error.

Add both versions of the design to a dummy cell. Assign a low-level color (e.g. PURPLE) to the dummy layer used by all components of the second version. The missing components will then show through where they are not covered by the successfully snapped components. You can then add the missing components by hand, carefully considering how to snap each coordinate to grid.

Conclusion

You should read the complete description of the UnStream utility in the Layout Editor Reference Manual before using the utility to perform real work. The lessons in the tutorial have only scratched the surface of the additional control you have over how the translation proceeds. Stream files from various sources may have many non-standard features that require special handling on import.

The same manual contains a complete description of the STREAM command used to create stream files. Also read about the STREAM parameters of the LAYER command that control how data on specific layers is translated into stream format data.

You should also read the additional information in the following files for special import/export cases:

stream_file.ERR If the UnStream utility generates errors or warnings they will be stored in a file with this name. Read this file carefully and run the utility again after reading about the parameters that control the non-standard features found in your stream file.

stream_file.ALI This file is the alias file created by the first pass of the UnStream utility. It contains many comments to aid you in the customization of this file for cell names and layer correspondences for specific stream data types and text types. This file can also be edited to create a reusable layer mapping file.

Q:\ICWIN\DOC\ARRAYS.TXT If your stream file was created by Cadence software tools, it may contain non-standard array definitions. Read this file to learn how to accommodate these types of stream files.

Q:\ICWIN\DOC\SFMAP.TXT This file contains complete instructions for the SFMAP utility used to convert the structure names in a stream file. This utility is the only method available to create lower case or mixed case structure names in stream files created by ICED™.

Creating Useful Command Files

Command files are files of ICED™ commands. The commands in the file are executed as though they were typed at the keyboard. In other words, the commands are interpreted one command at a time. However, you can execute an entire command file with one keystroke.

Command files can be very simple. The startup command file used to define the environment of new cells is a simple command file. It contains basic editor commands that set layer properties, etc.

However, command files can get quite complex. They can add or modify geometry, manipulate strings and mathematical expressions, perform conditional blocks of commands and loops, export data, and even execute outside compiled programs or DOS commands.

If you have done the preceding tutorials, you have already executed several command files aside from startup command files. The ED.CMD file (page 37), the SHOW1.CMD file (page 106), and the DEEPSHOW.CMD file (page 157) are all so useful we have already used them in the more basic tutorials.

The lessons and overviews in this tutorial are intended only to introduce you to the concept of command files. You will be creating several useful command files, but you will not be a command file programming expert until you read all of the additional information in the Command File Programmer's Reference Manual.

Locations of Command Files

The three command files mentioned above are all technology and project independent. They could be useful in any design. They are all located together with other **general-purpose command files** in the **Q:\ICWIN[33]\AUXIL** directory. This directory is on the search path for command files, so any command files in this directory can be executed without having to supply the path to the file.

[33] Remember that Q: and \ICWIN are used throughout the manual to represent the drive and directory where you have installed the ICED™ software.

Some command files are tailored for specific projects or technologies. Startup command files are an example. These **technology-specific command files** should be stored separately so the correct version for your technology is always used. Store these types of command files in an appropriately named **subdirectory of the Q:\ICWIN\TECH** directory. The installation provides examples of several technology-related command files in the Q:\ICWIN\TECH\SAMPLES directory. If you want to use one of these, copy the file to your own technology directory and edit the values in it for your technology.

For example, the CONT.CMD command file creates an array of contacts with minimal user interaction. However, technology-specific values such as the minimum size of each contact and the minimum distance between contacts should be customized in the file before you use it in your design.

You can store command files in other locations. We recommend against placing them in any of the other existing directories created during the installation or in cell libraries. However, you can create other directories for your command files. We will show you how to add your new directory to the search path for command files in one of the following lessons.

Using a Command File

You need to open the layout editor to execute a command file. Open an appropriate console window now with the ICED icon on your desktop. We will create a new subdirectory of the TUTOR directory for this tutorial. Type the DOS commands below in the console window to create the directory, make it the current directory, and open the editor.

```
CD TUTOR
MD CMDFILES
CD CMDFILES
ICWIN TESTCMD
```

There are three ways to execute a command file:

Menu option 3:**@%.cmd** This method lists command files in each directory on the command file search path, allowing you to select a command file from the menus.

Type the *@file_name* command This method can be used to execute any command file, including those not on the search path, when you include the directory path with the command file name. When the command file is in a directory on the search path, you can omit the directory path in the *@file_name* command.

Pressing a key When you assign a *@file_name* command to a key combination with the KEY command, the command file will execute when you press the key combination.

We will concentrate on the last two methods in this tutorial, but let us look at the first method briefly. Select the menu option:

3:**@%.cmd**

If there were command files in the current working directory, they would be listed in the first menu list. The current directory is always the first directory in the command file search path. However, since we are using an empty working directory, the files in the sample technology directory are listed first. This directory was added to the command file search path by the project batch file (Q:\ICWIN\ICWIN.BAT) used to open the layout editor. To look at the files in the next directory on the search path, select:

NextPATH

Now the command files in Q:\ICWIN\AUXIL are listed. Not all of the command files in this directory are listed. If you use another method to look at all the files in the directory, you will see several files with names similar to _*name*.cmd. Command files with names that begin with '_' are "helper" command files intended to be called within other command files to perform a task. They are not designed to be called directly from the editor, so they are not included in the list built by the menus. We will be using some of these "helper" command files in a later lesson.

Note that the ED, DEEPSHOW, and SHOW1 command files are all in this list. Choose instead the following menu item:

COLORS

The command file Q:\ICWIN\AUXIL\COLORS.CMD is executed. Note the success message left on the screen below the prompt line by the command file. The result of the last command in the file is always left on the screen. This is often the only proof you have that anything was actually done by a command file. If you need to see exactly what a command file did, look at the journal file. Do this now with the menu options:

1:**FILE → edit.JOU**

A Notepad window is opened allowing you to view the journal file that records every command executed in the current session. The commands in the startup command file come first. Scroll down the file until you see a line similar to:

```
        ! @Q:\ICWIN\AUXIL\COLORS.CMD
```

This line is a comment that was added to the file to indicate that the command file was executed. The lines that follow are the commands that were executed by the command file. When the command file completed, the following line was added to the journal file:

```
        ! Exit file @Q:\ICWIN\AUXIL\COLORS.CMD
```

This method is very useful for debugging your command files. You can see exactly what actions were performed by your command file. (When a LOG command is included at the start of a command file, it controls how much information is added to the journal file.)

You must close this Notepad window to continue work in the layout editor. Close the Notepad window now with the 'X' button in the upper right corner of the window.

Creating a Command File from Menu Choices

The journal file is useful not only for looking at the results of a command file, but also for creating them. In this lesson we will execute some layout editor commands, then collect those commands into a new command file that will execute all of the commands at once. Using this method, you do not even need to know the syntax of typed commands to create a command file.

First create a box on layer 1. (If you have used the sample startup command file and created a new empty cell, this layer should have the name NWEL and it should already be the default layer. But, just in case, we'll select the layer by number so that you are definitely adding a box on layer number 1.)

1:**UseLay → By # → 1:NWEL → BOX**

Now we will swap all geometry on layer 1 to layer 2. Remember that whenever you use the SWAP command, you must be sure that only the components you want to swap are selected, so we should begin the process by unselecting all components.

1:(UNSEL)**all**

1:(SELECT)**layer → By # → 1:NWEL → Return → ALL**

2:(SWAP)**layers → By # → 1:NWEL → By # → 2:NDIF**

1:(UNSEL)**all**

Now open the journal file in a Notepad window with the following menu option.

1:**FILE → edit.JOU**

You want to be careful to never edit and then save the journal file during a normal edit session. If you do, you will not be able to recover the work you've done if your session is terminated abnormally. So the first step we will do is to save this file to a new name before we edit it. In the Notepad window, use the menu options:

File → Save As

In the "Save as type" box, use the drop-down list to select "All Files" or else the extension ".txt" might be added to your file name. In the "File name" box, type the name:

SWAP12.CMD

Then click the "save" button.

Now delete all of the lines from the top of the file down to the ADD BOX command near the bottom of the file. (This includes deleting the ADD BOX command. We don't want that command in our command file.) Delete the final comment line as well. Now the only lines left are as follows:

```
UNSELECT  ALL ! Select count=0
SELECT  LAYER NWEL  ALL ! Select count=2
SWAP LAYERS NWEL AND NDIF
UNSELECT  ALL ! Select count=0
```

The '!' character indicates the start of a comment. Delete the comments from the first, second and fourth lines. (If you have used a different startup command file and your layer names are different than those above, don't be alarmed. You can replace the layer names with layer numbers if you might want to use this command file for real work at a later date. If not, just leave whatever layer names are shown.)

Add the following line to the bottom to of the file so that your command file leaves a success message at the bottom of the screen when it executes. Lines with a '$' prefix leave comments on the screen.

$ SWAP12.CMD executed successfully.

The entire file now looks like:

```
UNSELECT  ALL
SELECT  LAYER NWEL  ALL
SWAP LAYERS NWEL AND NDIF
UNSELECT  ALL
$ SWAP12.CMD executed successfully.
```

Now **save the file and exit** to return to the layout editor. You can select this command file with the menus as we did with the COLORS.CMD file we used above. However, it is often easier to simply type the *@file_name* command that calls a command file. You do not even need to include the ".CMD" file extension. Type the following:

@SWAP12

When you execute this command file something strange happens. The near cursor appears and a command that begins with XSELECT is shown on the command line. This rather unsettling behavior is the result of a common problem in command files that deal with selected components. When the SELECT LAYER NWEL ALL command was executed, nothing was selected because there are no longer any components on layer NWEL. When the SWAP command was executed, no components were selected, so the editor generated an embedded select command to allow you to select something for the swap.

This type of behavior would be puzzling and awkward for other users of your command file who expect this command file to work without any user interaction. It would be better if the SWAP command did nothing when there are no components on layer 1 to swap.

This is easily accomplished by adding a line to your command file. Reopen the file by launching Notepad outside of the editor. You can do this by opening a second console window with the ICED icon on your desktop (or by using the 3:SPAWN menu option) and then typing the following command in the new window:

NOTEPAD TUTOR\CMDFILES\SWAP12.CMD

In the Notepad window, add the following line as the first line in the file. This command (used only in command files) disables the embedded select feature for all commands that execute on selected components. We strongly recommend adding this command to command files that operate on selected components.

XSELECT OFF

Save the file and close Notepad. In the layout editor, execute the command file again:

@SWAP12

This time the file runs to completion with no mysterious errors even though it does not swap anything. Add another box on layer 1 and rerun the command file to see it successfully swap the layer of a component.

Close the layout editor once you have experimented with your new command file.

1:**FILE → LEAVE**

Creating a New Command File Directory

A general purpose command file like SWAP12.CMD is best stored with the other general-purpose command files in the Q:\ICWIN\AUXIL directory. However, suppose that you are so delighted with your new talent (successful command file programming) that you intend to write a lot of new command files. You can leave the directory of command files supplied with the installation intact and create a new directory for your new files.

Create a new subdirectory of our tutorial directory and move your new file there. You can use any method to accomplish this. To use console commands, switch to the second console window (where the Q:\ICWIN directory is still the current directory) and type:

```
MD  MYCMDFILES
MOVE  TUTOR\CMDFILES\SWAP12.CMD  MYCMDFILES
```

To add this directory to the search path for command files, we must edit the project batch file. It is best to leave the batch file supplied with the installation intact, so we will copy this file before we edit it. Use whatever method you like to accomplish this. To use console commands, type in the second console window:

```
COPY  ICWIN.BAT  MYICWIN.BAT
NOTEPAD  MYICWIN.BAT
```

In the copied batch file, find the line that stores the command file search path in an environment variable. It should look like the following:

```
set iced_cmd_path=q:\icwin\tech\samples;
```

Add the path to your new directory after the semicolon (;). The line should now look similar to:

set iced_cmd_path=q:\icwin\tech\samples;**q:\icwin\mycmdfiles;**

We will make one other change to this project batch file that will allow you to see the startup messages displayed by the editor before it opens the layout view. The pause=*seconds* option on the editor command line controls how long these messages are displayed before the window is replaced with the layout view. Find this option on the command line near the bottom of the file, and change the value to a 10 second wait.

q:\icwin\iced %1 … pause=**10** …

Save the file and close the Notepad window.

Now return to the first console window. The current directory should still be the tutorial directory Q:\ICWIN\TUTOR\CMDFILES. Open the editor with the modified project batch file.

MYICWIN TESTCMD[34]

Note that some console messages are now displayed and that a countdown is going on at the bottom of the window. Press the **<Esc> key** before the countdown gets to 0. This allows you to look at the messages for as long as required.

Note the lines that begin with "! CMD directory *n*:". These are the directories in the command file search path. The current directory is listed first. Then come the directories in the environment variable defined in the batch file. Note that your new directory is included. Finally, the Q:\ICWIN\AUXIL directory comes last. This directory is always added to the end of the command file search path.

To clear these messages and open the layout view, press **<Enter>** or both mouse buttons.

Now that the editor is open, let us verify that the editor can find your command file. Type:

@SWAP12

[34] If you get a message from the operating system about a lack of environment space while executing this batch file, you need to increase the amount of memory available for environment variables. If you do not know how to do this (the method varies depending on the operating system,) contact IC Editors technical support.

Assigning a Command File to a Function Key

You can use the KEY command to assign any command line to a function key or to a combination of a function key and one of the keys, <Alt>, <Ctrl>, or <Shift>. In this lesson, we will assign the command that calls your command file to the key combination <Shift> and <F12>.

Type the following on the command line:

KEY SF12 @SWAP12

Now press and hold the <Shift> key down while you press the <F12> key. Note that the command file is executed. Now close the layout editor with:

1:**FILE → EXIT**

This key assignment is stored with the cell file. When you reopen this same cell at a later date, this KEY definition would still be in place. But suppose that you want this KEY definition made in all new cells. Then you must make the key assignment in the startup command file.

Change to the sample technology directory and copy the sample startup command file. Then open the copied file with a text editor. To accomplish this with console commands, type in the console window:

CD \ICWIN\TECH\SAMPLES

COPY NEW.CMD MYNEW.CMD

NOTEPAD MYNEW.CMD

At the bottom of the file are the existing key assignments. Add your definition as the last line of the file. (Type the command using the syntax you learned above. We will explain the syntax used by the other key definitions in the following lessons.)

KEY SF12 @SWAP12

Save the file and close the Notepad window. Now you must edit the project batch file to use this new copy of the startup command file. In the console window, type:

CD \ICWIN

NOTEPAD MYICWIN.BAT

Change the editor command line option that sets the startup command file to be similar to the following:

q:\icwin\iced %1 … start=q:\icwin\tech\samples\mynew …

Save the file and close the Notepad window. In the console window, change the current directory back to our tutorial directory and use the modified batch file to open the editor to create a new cell.

```
CD TUTOR\CMDFILES
MYICWIN NESTCELL
```

Once the view window is open, note that your new line in the startup command file is reflected on the history line below the command prompt. The key definition is reported in a different syntax, but the result is the same. Test the key definition by pressing the keys <Shift><F12>.

Overview of ICED™ Macros

When you executed the KEY command in the last lesson, you created a special case of an object called a macro. A macro is simply a stored string of ASCII characters. In the case above, you created a macro with the name KEY.SF12. The string "@SWAP12" is stored as the value of the macro very similarly to the way you would store a string in a variable in any common programming language.

The KEY.SF12 macro is a **user macro**. User macros are created and defined by commands in the editor. They can have their contents altered by assigning new values to them, and they can be deleted entirely with the REMOVE command.

A second type of macro is a **system macro**. This type of macro is defined by the ICED™ program. You cannot change the value directly, and you cannot delete a system macro. They contain information about your design and environment settings.

To see the entire list of macros defined in your current cell, type the following command on the command line:

```
SHOW MACROS
```

You can see that you already have a long list of macros defined in your cell. The user macros are listed first. Scroll back to the top of the list with the scroll bar on the right side of the window. Note that your KEY.SF12 macro is shown.

Look briefly at the long list of system macros. You can use all of these macros in your command files. We will be using a few in later lessons. Near the end of the list is the USE.LAYER macro. This macro saves the layer number of the current default layer. Every time you change the default layer with the 1:**UseLay** menu option, the new value is stored in the system macro automatically. Return now to the layout view by pressing **<Enter>**.

Whenever you define or use a macro in a command file, you should precede the macro name with one of the two special characters below.

#	Used to indicate that you are referring to the name of the macro.
%	Used to indicate that you are referring to the contents of the macro. The string "%*macro_name*" will be replaced with the string stored in the macro.

For example, if you have a macro with the name MYMAC, you could write the following line in a command file:

#MYMAC = %USE.LAYER

Let us assume that the USE.LAYER system macro currently contains the string "1". Before the line above is executed, the contents of the USE.LAYER macro will replace the macro reference %USE.LAYER. The line will be interpreted as:

#MYMAC = 1

When the line is executed, the string "1" will be stored in the macro MYMAC. While the value of a macro is always a string, when a command is executed, the string can be interpreted as a number (or even a position). For example, once MYMAC is defined, the following is a valid command:

ADD BOX LAYER= %MYMAC

In some cases, the '#' or '%' is optional, but it is a highly recommended programming practice to always precede a macro name with one symbol or the other to avoid confusing mistakes in a command file.

You can use macro references to build long strings. When you need to separate a macro reference from other characters without a space use the '^' character.

#MYMAC = %TMP^%CELL^.TXT

The system macro TMP contains the path to the directory where temporary files are stored. The system macro CELL contains the name of the cell you are currently editing. If TMP currently contains the string "Q:\ICWIN\TMP\" and CELL contains "MYCELL", then the line above will be interpreted as:

#MYMAC = Q:\ICWIN\TMP\MYCELL.TXT

If you need the equivalent of an array of macros, you can even use string substitution on the left side of the '=' in a macro assignment statement.

```
#COUNTER = 1
#MYMAC.%COUNTER = X
```

The second line will be interpreted as:

```
#MYMAC.1 = X
```

Defining User Macros

Before you define a macro, you must decide on its scope. **Scope** refers to how long a macro exists, and whether or not it can be used outside of the command file in which it is defined. Macros have either local or global scope.

Local macros	can be used only by the command file that defines them. They will be deleted automatically at the end of the command file.
Global macros	persist after the completion of the command file. They can be used in other command files, and also in editor commands outside of any command file.

If a command file defines a local macro with the same name as an existing global macro, the local macro definition will be used and the global macro is ignored. However, if you add the keyword **DEFAULT** to the macro definition, then if a macro with that name already exists, the new definition is ignored and the command file will use the existing macro definition.

There are a few rules to macro definition:

- Macros must be defined before they can be used.
- Use DEFAULT as the first keyword in the macro definition when you want the definition to be ignored if a macro with that name already exists.
- Either the LOCAL or GLOBAL keyword should be used in the macro definition to determine the scope of the macro.
- Macro names are case independent and they can use up to 32 characters. In addition to alphanumeric characters, the characters '$', '#', '.', or '_' can be used in macro names. (Do not use '#' as the first character in a macro name.)
- Each macro must be defined with an initial value. When you omit the macro value, or define it with the $*response_type* option (you'll see some examples of this in the following lessons), the user of the command file will be prompted to supply the value of the macro

- User macros (except for KEY macros) are never saved in the cell file. Once you terminate the editor, even global macros are deleted automatically.

You can define macros outside of a command file. In this case, the macro must be global. Create a macro now by typing the command:

GLOBAL #MYMAC = 5

Now display all currently defined user macros with the command:

SHOW USER

Note that your KEY.SF12 macro is still defined along with the new macro. Return to the layout view by pressing **<Enter>**.

Now delete the new macro with the command:

REMOVE #MYMAC

Creating a Command File That Uses Macros

Now we will write a command file that will create some geometry in the cell. This command file will add a polygon in the shape of the digit '0'. The size, layer, and position of the polygon will all be stored in macros.

The geometry will actually be created by a command file that already exists in your ICED™ installation, _char0.cmd. (This command file was created as a "helper" command file used by Q:\ICWIN\TECH\SAMPLES\SERIAL.CMD.) The only command in this command file is shown in Figure 71.

```
ADD  POLYGON  LAYER=%layer  OFFSET=%off  AT       &
          {%fmag*(0.35, 0.2)} {%fmag*(0.25, 0.2)} {%fmag*(0.25, 0.25)}        &
          {%fmag*(0.45, 0.45)} {%fmag*(0.5, 0.45)} {%fmag*(0.5, 0.25)}        &
          {%fmag*(0.45, 0.2)} {%fmag*(0.35, 0.2)} {%fmag*(0.35, 0.0)}         &
          {%fmag*(0.55, 0.0)} {%fmag*(0.7, 0.15)} {%fmag*(0.7, 0.85)}         &
          {%fmag*(0.55, 1.0)} {%fmag*(0.35, 1.0)} {%fmag*(0.35, 0.8)}         &
          {%fmag*(0.45, 0.8)} {%fmag*(0.45, 0.75)} {%fmag*(0.25, 0.55)}       &
          {%fmag*(0.2, 0.55)} {%fmag*(0.2, 0.75)} {%fmag*(0.25, 0.8)}         &
          {%fmag*(0.35, 0.8)} {%fmag*(0.35, 1.0)} {%fmag*(0.15, 1.0)}         &
          {%fmag*(0.0, 0.85)} {%fmag*(0.0, 0.15)} {%fmag*(0.15, 0.0)}         &
          {%fmag*(0.35, 0.0)} {%fmag*(0.35, 0.2)}
```

Figure 71: Q:\ICWIN[35]\AUXIL_CHAR0.CMD

The ‘&’ character is a continuation character that allows a command to continue on successive lines. All of these lines are combined to form a single ADD command.

We must define 3 macros to use this command file:

layer	used to supply the layer id in the ADD command.
off	used to define the base location of the polygon. This coordinate pair is added to each of the positions in the ADD command.
fmag	used to set the height of the polygon. Each coordinate will be multiplied by this real number to scale the polygon’s size.

Each coordinate definition in the command file is defined with an expression indicated with a pair of braces {}. We will explore expressions more in a later lesson. Basically, the expression in {} is evaluated before the command is executed. In this case, the multiplication symbol * indicates that each coordinate will be multiplied by the number stored in the fmag macro.

We will create our new command file to define these three macros and then call the _char0.cmd file to create the polygon.

First, we will create a simple version of this command file that simply defines each macro with a value. Later we will improve it to prompt the user to supply each value. Finally we will add some value checking to insure that the user really enters valid values for all of the macros. Value checking is always a good idea. For example, if the fmag macro contained a string that could not be interpreted as a number, the _char0.cmd command file would fail.

[35] Remember that Q: and \ICWIN are used throughout the manual to represent the drive and directory where you have installed the ICED™ software.

Open a text editor to create the command file in the MYCMDFILES directory. The name of the file should be “digit0.cmd”. You can type the following commands in the second console window to accomplish this:

CD MYCMDFILES

NOTEPAD DIGIT0.CMD

We will define the macros with global scope so that they will be available to the command file _char0.cmd. If we defined them with the LOCAL keyword instead, the helper command file would not be able to access them and would fail with an error message.

Type the first macro definition in the notepad window as follows:

GLOBAL #layer = %USE.LAYER

Remember that the USE.LAYER macro is a system macro that stores the current default layer. The line above defines a macro with the name “layer” and stores the number of the current default layer as its value.

The next two lines define the other two macros and store appropriate values for each of them.

GLOBAL #fmag = 10

GLOBAL #off = (0,0)

The next line in the command file should call the _char0.cmd file:

@_char0

Since this command file is stored in a directory on the command file search path, you do not need to supply the path to the file.

As the final line of the file, add a $*comment* line that leaves a success message
on the prompt line. The system macro EXEC.FILE stores the name of the currently executing command file.

$ %EXEC.FILE executed successfully.

Save the file (but leave the text editor open) and execute your new command file in the layout editor with the command:

@digit0

You may need to change the view window to see that a polygon 10 units high in the shape of a ‘0’ has been added on the default layer at (0,0).

Prompting the User for Macro Values

The digit0.cmd command file would be more useful if the user could supply the layer, height, and position of the polygon when the command file is executed. This is easily accomplished with the use of the $PROMPT keyword in each macro definition.

Return to the text editor and change the first macro definition to the following:

GLOBAL #layer = $PROMPT = "layer for digit polygon"

When this command is executed, the prompt string “layer for digit polygon” will appear below the layout editor command line and the editor will wait until the user types something and presses <Enter> to continue. Whatever the user types before the <Enter> will be stored as the value of the macro.

Change the next macro definition to a user prompt in the same way.

GLOBAL #fmag = $PROMPT = "height of digit"

The final prompt will be for the base coordinate of the new polygon. For this macro we could use the same method and have the user type the coordinate pair with the keyboard. However, we will use a different method and have the user digitize the coordinate pair with the mouse by adding the keyword “POSITION” to the end of the macro definition. (The list of keywords that require the user to define a position (or a list of positions) with the mouse are shown in Figure 72.)

GLOBAL #off = $PROMPT= "define position with mouse" POSITION

When the line above is executed, the indicated prompt string will appear, and the editor will wait until the user presses the left mouse button to continue. (You can even omit the $PROMPT=”*string*” parameter, and a default prompt string will be used.) The coordinates of the position under the cursor when the button is pressed will be stored as the value of the macro.

Now **save the file** (but leave the text editor open) and execute the modified command file in the layout editor.

@digit0

When prompted for the layer, you can type either a layer number or a layer name. When prompted for the height, enter a number. When prompted for the coordinates, move the mouse to a convenient location and press the left mouse button. Look at the polygon that is created.

POSITION
BOX
POLYGON
DISP
X_DISP
Y_DISP
NEAR

Figure 72: Macro definition mouse prompt keywords

Using the Macro Verification Command Files _get_xxx.cmd

Execute the digit0.cmd command file again. (Use the 1:**Again** menu option to repeat the last command.) This time, type something that is not a layer name or number when prompted for the layer. You can see that the ADD command in _char0.cmd fails in this case.

When command files fail due to bad macro values, the error messages can be very mysterious. It would be better to insure that the user has entered a valid layer string before using the macro value.

There are several command files supplied with the ICED™ installation that prompt the user for different types of macro values and verify that the user response is appropriate. For example, when you use the command file _get_lay.cmd, you can be sure that the user has typed a valid layer name or number before using the macro value in the rest of your command file.

_get_ans.cmd	User must type a single character
_get_dev.cmd	User must type a valid device or port name
_get_int.cmd	User must type a integer
_get_lay.cmd	User must type a valid layer name or number
_get_real.cmd	User must type a real number

Figure 73:Macro verification command files

These command files use default local macros to define the prompt string and default value. You can override these local values when you call the command file. To override a default local macro definition, you can define a local macro with the same name on the command line used to call the command file. Then this macro definition is executed as though it is the first line of the command file. The default local macro definition in the command file is then not executed since the macro already exists.

Return to the text editor again and add the following three lines at the very top of the command file:

```
@_GET_LAY;                                          &
        LOCAL #prompt = "layer for digit polygon";  &
        LOCAL #default = %USE.LAYER
```

The semicolons (;) and ampersands (&) are critical. Remember that the ampersands allow a single command to be spread over several lines. The macro definitions must be on the same command line as the call to _get_lay.cmd. The semicolons separate the individual commands on the line. You will get syntax errors if you omit them.

The macro default defines a default value that will be used if the user simply presses <Enter> in response to the prompt. The command file redefines the actual prompt string using the value of the macro default (if supplied) and the string you supplied in the prompt macro as follows:

```
#prompt="Enter %prompt [%default]:"
```

So if the current default layer number is 1, then the prompt used by _get_lay.cmd will be:

“Enter layer for digit polygon [1]:”

If the user presses simply <Enter>, then “1” is stored as the valid layer string. If the user types some response that is not a valid layer name or number, then _get_lay.cmd will prompt the user with “Invalid layer name or number” and the user must press <Enter> to clear this prompt. Then the user is prompted for the layer again with the prompt above. When the user types a valid layer name or number, than that is stored as the layer string and the _get_lay.cmd command file terminates.

For all of these macro verification command files, the validated user response is stored in the global macro ret.value. So you must store this response in a new macro before calling another _get_*xxx*.cmd file. Otherwise the response will be overwritten before you use it.

To store the user response in the macro where we need it, change the definition of the layer macro to:

```
GLOBAL #layer = %ret.value
```

Similarly, change the definition of the fmag macro to:

```
@_GET_REAL; &
        LOCAL #prompt = "height of digit"; &
        LOCAL #default = %LAYER.WIRE.WIDTH.%layer
GLOBAL #fmag = %ret.value
```

You may ask about that strange macro reference, % LAYER.WIRE.WIDTH.%layer. There are several system macros for each layer. One of these is the LAYER.WIRE.WIDTH.* macro, where * is replaced with the layer name or number. Since macro string substitution works from right to left, first the %layer macro reference will be replaced with the value of the macro. If the layer macro is defined with the value “M1”, then the macro reference resolves to the system macro, LAYER.WIRE.WIDTH.M1. The value stored in this system macro is then stored in the macro default and is used to set the default height of the digit polygon.

We will leave the definition of the off macro as it is. This macro needs no validation since it was not typed by the user, but defined with the mouse. The macro will definitely contain a valid coordinate pair.

The entire file digit0.cmd should now look like the following:

```
@_GET_LAY;                                           &
        LOCAL #prompt = "layer for digit polygon";   &
        LOCAL #default = %USE.LAYER
GLOBAL #layer = %ret.value

@_GET_REAL;                                          &
        LOCAL #prompt = "height of digit";           &
        LOCAL #default = %LAYER.WIRE.WIDTH.%layer
GLOBAL #fmag = %ret.value

GLOBAL #off = $PROMPT= "define position with mouse" POSITION

@_char0

$ %EXEC.FILE executed successfully.
```

Save the command file and execute it a few times with different responses. Use the default values for one test. Change the default layer with 1:**UseLay** menu option, then execute the command file again and see that the prompts reflect the new default layer number and height. Try to enter invalid responses and see what happens.

Overview of Mathematical and Boolean Expressions

An expression in a command file is a string surrounded by curly braces, {}. The string is evaluated and replaced with the result of the expression before the command is interpreted by the editor.

We have already seen an example of a mathematical expression in a command file. The _char0.cmd command file contained several expressions similar to:

```
{%fmag*(0.35, 0.2)}
```

In this case, if the value stored in the fmag macro is a number, then each coordinate will be multiplied by the number and the resulting coordinate pair is used in the command. If fmag = 2, then the expression will be replaced with the following coordinate pair before the command is executed:

```
(0.7, 0.4)
```

However, if the curly braces were not present, the command interpreter would perform no expression evaluation. The macro substitution would still be performed, but the editor would try to execute the ADD command with the following string instead of a simple coordinate pair and the command would fail with a syntax error:

```
2*(0.35, 0.2)
```

The curly braces are required any time an expression requires more evaluation than simple macro string substitution. This includes all mathematical expressions, Boolean expressions, and function calls. (We will learn more about function calls in a later lesson.)

For example, if you need to calculate how wide a wire needs to be, you might use a statement similar to:

```
#wire_width = {%pitch - %min_dist}
```

As long as the pitch and min_dist macros contain real numbers, then the real number result of the expression is stored as the value of the wire_width macro. For example if pitch = 3 and min_dist = 1, then "2" would be stored as the value of wire_width. However, if the curly braces were not used, then the string "3 - 1" would be stored as the value of the macro. This would probably cause an error later on in your command file when you tried to use the value of wire_width in a command.

+
-
*
/

Figure 74: Mathematical Operators

Boolean expressions evaluate to a single number. This number is interpreted by the editor as one of the two valid Boolean values: TRUE or FALSE.

FALSE = 0

TRUE = 1 or any non-zero number

Boolean values can be used by IF, ELSEIF, and WHILE statements to control the execution of blocks of commands. All of the operations shown in Figure 75 will compare numbers ("==" and "!=" also work on coordinate pairs) and evaluate to a Boolean value when used in an expression. For example, you might write the following Boolean expression and then use it in an IF statement:

```
#done = {%counter <= 10}
IF (%done) …
```

<
<=
==
!=
>=
>

Figure 75: Number Comparison Operators

In this case, the done macro will contain the value TRUE if the counter macro contains a number greater than or equal to 10. Done will be set to FALSE if counter is less than 10. The code following the IF statement will be executed only when done contains TRUE.

You can also combine Boolean values with the Boolean operators shown in Figure 76. If you are combining several Boolean operations, you can use the & continuation character to make the statement easier to read. Remember to surround the entire expression with curly braces or the expression will not be evaluated.

```
#error = {%x_coord < 0        ||  &
          % x_coord > 10000   ||  &
          % y_coord < 0       ||  &
          % y_coord > 10000        }
```

&&	Boolean AND
||	Boolean OR

Figure 76: Boolean Operators

In this case the macro error will be set to TRUE if either of the coordinate macros is outside of the range 0:10,000.

There is no Boolean NOT operator. If you need to write a conditional statement that tests if a Boolean value is FALSE, use the == operator. For example, let us assume that the code after the following IF statement should be executed only when the value of the error macro is FALSE.

```
IF (%error == 0) …
```

Conditional Blocks of Commands

We have already mentioned the IF, ELSEIF and WHILE commands. The IF and ELSEIF commands can control either a single statement or a block of statements surrounded with curly braces. The WHILE command is used to execute a block of commands in curly braces repeatedly until the Boolean value that controls the loop is FALSE.

For all three commands, the Boolean expression that controls the execution is contained in parentheses. Since ICED™ expects an expression in this context, curly braces are not required to force evaluation.

For example, if you need to execute a single $*comment* statement when the macro error is FALSE, you would write it this way:

```
IF (%error == 0) $ error is FALSE
```

When you need to execute a block of commands, add the open curly brace on the same line as the IF statement, and follow the last command in the block with the closing curly brace. For example, if you needed to execute a block of commands when error is TRUE, then you could write the following:

```
IF (%error) {
        @error_handler.cmd
        $ error is TRUE
        RETURN    ! immediately end execution of the command file
        }
```

You could combine both of these IF statements along with an ELSEIF statement as shown in the following:

```
IF (%error == 0)               $ error is FALSE
ELSEIF (%continue_on_error)    $ error is TRUE, but continue anyway
ELSE                           {
        @error_handler.cmd
        $ error is TRUE
        RETURN                 ! end execution of the command file
        }
```

Using a WHILE Loop to Copy Selected Components

The WHILE command will continue to execute a block of commands repeatedly as long as the Boolean expression is TRUE. Whenever you use a WHILE statement you must use caution to insure that the Boolean expression will eventually become FALSE, or the block of commands will execute forever. This is referred to as an "infinite loop". When a block of commands that includes no user interaction commands is looping infinitely, the only way to interrupt the command file is to close the editor entirely using an operating system method (e.g. the X button in the upper right corner of the window.) Your cell files will not be saved, and you must recover your work with an edited journal file.[36] For this reason, always debug command files with WHILE loops in temporary test cells so that you will not lose any work.

One common type of WHILE loop uses a counter to control how many times a loop will be executed. Let us create a command file now that uses such a WHILE loop to copy selected components multiple times. (This is roughly the equivalent of creating a group, creating an array of the group, then ungrouping all components.)

Create and open a text file with the name NX_COPY.CMD in the MYCMDFILES directory. To use a console command to accomplish this, type the following in the second console window:

```
NOTEPAD NX_COPY.CMD
```

First we will create a WHILE loop that insures that at least one component is selected. Since we use this loop, this command file will not require the XSELECT OFF command usually found in command files that operate on selected components. The command file will not continue until at least 1 component is selected.

Type the following in the Notepad window:

```
WHILE(%n.select==0){
    PROMPT "Select components to copy.[Both buttons to cancel]"
    SELECT IN
}
```

This is an extremely flexible loop for selecting components. The system macro n.select always contains the number of currently selected components. If components are already selected when this command file executes, the Boolean

[36] The journal file must be edited to remove all lines from the end of the file up to the call of the unfortunate command file. The next time ICED™ is opened to edit the same cell, it will ask you if you want to recover your work. When you reply with a <Y>, then the edited journal file is executed, recovering all of your work done prior to the call of the command file.

expression that controls the loop will be FALSE and the block of statements within the curly braces will not be executed at all.

When no components are selected, the n.select will contain “0”, the Boolean expression will be TRUE, and the statements in the block will be executed. The PROMPT command will leave the indicated string displayed as the prompt on the command line when the SELECT IN command is executed.

If components are selected when the user of the command file digitizes the corners of the selection rectangle, then n.select is set to a number greater than 0, and the Boolean expression becomes FALSE. The block of statements is not executed again. However, if the user digitizes the selection rectangle such that no components are selected, n.select remains 0, the Boolean expression is still TRUE and the block of statements is executed again.

The net result of this simple loop is that the command file will not continue until at least one component is selected. If the user has already selected components for the copy operation, the command file will continue with no user interaction. The loop will not execute infinitely since the user is given the chance to end the loop and abort the command file by pressing both mouse buttons.

(The method of using an user interaction command in a WHILE loop is a good way to debug WHILE loops. When you are creating other WHILE loops in future command files, it is a good programming practice to add some sort of a user interaction command in the loop so that the command file displays some message and waits for a user response before continuing. If your loop is not progressing correctly, you can cancel the command file with both mouse buttons any time the command file is waiting for user input. If your loop does not already contain a user interaction command, you can add a combination of a $*comment* and a PAUSE command. The PAUSE command will cause the command file to pause with the $*comment* displayed on the screen. The user must hit a key for the command file to proceed. This can even allow you to see the values of macros referred to in the $*comment* each time the loop executes.)

Now type the following to define a local macro to hold the x-displacement value for the copy operation:

```
LOCAL #disp=$PROMPT="Use mouse to enter copy spacing" X_DISP
```

The X_DISP keyword in a macro definition statement allows the user to digitize two points with the mouse. The x-displacement between the points is stored as the value of the macro.

Now we will define the local macro copy.count to hold the number of copies to be made. Use the _GET_INT.CMD helper command file to insure that an integer greater than 0 and no larger than 1024 is entered by the user.

```
@_GET_INT;                          &
    LOCAL #min=1;                   &
    LOCAL #max=1024;                &
    LOCAL #prompt="number of copies"
LOCAL #copy.count=%ret.value;
```

The _GET_INT.CMD helper command file will prompt the user for the copy count and test the value. If the user types something that is not a number, the command file responds with a warning and prompts the user again. If the user enters a number less than 1, then the command file responds with a warning that includes the minimum and maximum values and prompts the user again.

The definition of the copy.count macro stores the validated user response in the macro.

Type the following line to store the last component id used by the editor. This value is stored in the system macro id.max (Ids are assigned to components in ascending order as ADD commands are executed. All components have a unique id number.) The reason for this statement will be made clear later on, but it must be executed before the next loop.

```
LOCAL #n0=%id.max
```

Define a local macro to store the counter for the loop.

```
LOCAL #i=0;
```

Now type the lines that comprise the WHILE loop to perform the copy operation. The loop should execute the COPY command copy.count times.

```
WHILE(%i<%copy.count){
    COPY X=%disp
    #i = %i+1     ! correct syntax is really #i = {%i+1}
}
```

You will be correcting that line with the comment about syntax, so you don’t really need to type the comment.

Now **save the file,** but leave the Notepad window open.

Switch to the layout editor window. If you do not have some components already created in the view window, add some boxes with the menu options:

1:(ADD)**box**

1:**Again**

1:**Again**

Now execute the command file by typing on the command line of the layout editor:

@NX_COPY

Select some geometry at the prompt. At the next prompt, digitize two points. Finally, type a <**5**> at the prompt for the number of copies. The copies are created base on your responses.

Note the puzzling comment left on the history line once the command file is complete:

```
!LOCAL #I="0+1+1+1+1+1"
```

This was the actual statement executed when the following macro assignment statement was interpreted the last time through the loop:

```
#i = %i+1          ! correct syntax is really #i = {%i+1}
```

Since no curly braces were included, the statement is interpreted as simple macro substitution in a string. The first time this statement was interpreted, it was executed as:

```
#i = 0+1
```

The second time through the loop it was interpreted as:

```
#i = 0+1+1
```

However, since an expression in a WHILE command is automatically evaluated, the string stored in the i macro was interpreted as a mathematical expression as the WHILE command executed. So the loop still executed correctly, even though the value in macro i was probably not what you expected.

In this case, the command file executed correctly despite the mistake, but other command files that use the counter macro in other statements may not. Correct the macro assignment statement to force expression evaluation. This will store a number in macro i instead of a string. Switch to the Notepad window and correct the line.

#i = **{**%i+1**}**

We will add one more line to the command file that will aid in “undoing” the results of the command file. Remember that the UNDO command will not undo the results of any commands executed in a command file. It would make this command file far more useful if the user could simply undo all of the results with a single DELETE command.

The COPY X command in the loop will always unselect the original components and leave the copied components selected. Note that only the last set of copied components is now selected after you executed the first draft of your command file. It would be better to leave all of the copied components selected when the command file is completed.

This is why you saved the last id number assigned before the execution of the loop. Now you can use that value saved in the macro n0 to select all of the components created in the loop. Type the following line at the end of the file in the Notepad window:

```
SELECT IDS AFTER %n0
```

Add the following line to leave a success comment on the history line after the command file is complete:

```
$ %exec.file made %i copies
```

Save the file and execute it again. Note that the macro i is now set correctly to a number as reported in the success message. Note that only the copies are selected when the command file is complete. Execute a DELETE command and see that this performs the equivalent of “undoing” the results of the entire command file.

```
1:DELETE
```

You can continue to experiment with the command file until you get bored. Then close the Notepad window.

Functions

There are many functions available in ICED™. To describe all of them in detail is beyond the scope of this tutorial. However, we will list them below, and you can look up details in the ICED™ Layout Editor Reference Manual if you desire.

Cate-gory	**Function name**	**Purpose**
Value validation	DEVICE_EXISTS	Test that printing device exists
	MACRO_EXISTS	Test if macro is defined
	STD_COORD	Format coordinate string
	VALID_INT	Test that string represents a valid integer
	VALID_REAL	Test that string represents a valid real number
	VALID_LAYER	Test that string is a valid layer name
	VALID_CELL_NAME	Test that string is a valid cell name
Mathematical operations	SQRT	Square root of number (use positive numbers only)
	INT	Integer part of real number
	ABS	Absolute value of single number or vector length of coordinate pair
	SIN	SINE of angle in degrees
	COS	COSINE of angle in degrees
	TAN	TANGENT of angle in degrees
	ATAN	ARCTANGENT of single value or coordinate pair
	MIN	Minimum of two single numbers
	MAX	Maximum of two single numbers
String ma-nipulation	CMP	Compare two strings, case independent
	XCMP	Compare two strings, case dependent
	CHAR	Returns *n*th character in string
	LEN	Find length of string

(Continued on next page.)

Coordinate manipulation	ROUND ROUND1 ROUND2	Round single coordinate or coordinate pair to resolution grid
	POS1	Returns first coordinate pair in list
	POS2	Returns second coordinate pair in list
	POSN	Returns *n*th coordinate pair in list
	X	X-coordinate of coordinate pair
	Y	Y-coordinate of coordinate pair
	X0	First x-coordinate of list of coordinate pairs
	Y0	First y-coordinate of list of coordinate pairs
	X1	Second x-coordinate of list of coordinate pairs
	Y1	Second y-coordinate of list of coordinate pairs
Cells	CELL	Provides cell table index given a cell name

Figure 77: ICED™ Functions

All functions use a similar syntax.

***function_name*(*arg1*)**

or

***function_name*(*arg1*, *arg2*)**

There are two syntax restrictions you must keep in mind when calling functions in a command file. These restrictions prevent from ICED™ from misinterpreting the function name as part of an ordinary string.

- **Never insert a blank between a function name and the '(' character.**
- **Surround any function call in a macro assignment with curly braces, '{}'s.**

The second restriction is not important if you are calling the function in a condition of an IF or WHILE command already surrounded by parentheses.

For example, the CMP function is used to compare two strings. It will return 0 if the two strings are exactly the same when case is not considered. (If you want the case of the strings to be considered, use the XCMP function instead.) The function returns a negative number if the first string would come before the second when the strings are sorted in alphabetical order. The return value will be greater than 0 when the second string would come first.

If you needed to verify that a string stored in a macro is not equal to the current cell name (stored in the system macro CELL), you could use the following lines in a command file.

```
LOCAL #temp_name = $PROMPT "Enter cell name"
LOCAL #match = {CMP(%temp_name, %CELL)}    !0 if strings match
IF (%match == 0) {
        $Cannot use current cell name
        PAUSE
}
```

You can combine two statements and eliminate the match macro. In this case, the curly braces are not required because the IF command will always force evaluation of any function calls in the condition string.

```
IF (CMP(%temp_name, %CELL) == 0) {
```

Getting and Manipulating Component Coordinate Data

There are a number of functions that manipulate coordinates. Two of these are the X and Y functions that return a single coordinate from a pair of coordinates. You can then manipulate each coordinate as required. Single coordinates can then be combined back into coordinate pairs for use by other commands. We will use these functions to take the coordinates of a box and translate them into appropriate coordinates to create a wire that travels around the perimeter of the box.

To get the coordinates of a box into macros so that we can manipulate them, we will use the ITEM command. This command creates several macros containing information about a selected component. Each new macro name will begin with the *item_name* specified in the ITEM command. The list of macros created by the ITEM command changes based on the component type. However, the list of positions used to create the component is always stored in macros with the names *item_name*.POS.1, *item_name*.POS.2, etc.

The ITEM command requires that exactly one component is selected. So our example will test this with the N.SELECT system macro. This macro contains the number of currently selected components.

We will use the SELECT BOX command in the command file to allow the user to select the box for the outline wire. However, unless there is a visible prompt on the screen when this SELECT command executes, the user might be confused about what to select, so we will use the PROMPT command. This command changes the command prompt visible during the next command. The PROMPT command is very useful for providing explanations about what the user should do when user interaction is required in command files.

Any time you use commands in your command file that require the user to use the cursor in the view window, it is critical to insure that the view mode is on. This mode is defined by the VIEW command. So we will add VIEW ON at the beginning of the command file. If the VIEW mode was off, changes in geometry or changes in select marks will not be reflected in the display. (If your command file requires no cursor interaction from the user, the command file will execute more quickly when you add VIEW OFF to the beginning of the command file. For many command files, the update of the display after every command accounts for most of the execution time.)

Open a new text editor window to create a command file OUTLINEBOX.CMD in the current directory. To do this with console commands, type in the second console window (where the MYCMDFILES directory is the current directory):

```
NOTEPAD OUTLINEBOX.CMD
```

Now type the following lines:

```
VIEW ON
UNSELECT ALL
WHILE (%N.SELECT != 1) {
        UNSELECT ALL
        PROMPT "Select a single box for outline wire"
        SELECT BOX NEAR
}
ITEM LOCAL #orig_box
LOCAL #pos1 = %orig_box.POS.1
LOCAL #pos2 = {X(%orig_box.POS.1)} , {Y(%orig_box.POS.2)}
LOCAL #pos3 = %orig_box.POS.2
LOCAL #pos4 = {X(%orig_box.POS.2)} , {Y(%orig_box.POS.1)}

ADD WIRE %pos1 %pos2 %pos3 %pos4 %pos1
```

Note that the calls to the X and Y functions are surrounded with curly braces to force the function calls to be evaluated. Note also that the ITEM command includes the keyword LOCAL. This command requires either the LOCAL or GLOBAL keyword to determine the scope of the macros created.

Save the file. Open the layout editor from the first console window with MYICWIN.BAT if it is not already open.

Create a few boxes and a few other components. Then change the default layer using the typed command shown below before executing the command file so that it is easier to see the new wires as well as the original boxes.

```
LAYER POLY
@OUTLINEBOX
```

Execute the command file a few times. Try to select more than one box at a time and see what happens. Try selecting a component other than a box.

If you want to see exactly what macros are created by the ITEM command, change the LOCAL to GLOBAL in the command file, execute it again, and then use the **SHOW USER** command in the editor to see the macros created.

The macros created by the ITEM command are user macros and they can have their values modified directly. You can use these modified ITEM macros to create a new component or replace the original component. You use a macro with the name add.item.*item_name*. This macro contains an ADD command string that will create a component from the current information stored in the item_name macros. We will demonstrate this method in the lesson on page 262.

Looping Through All Subcells

Two command files are included with the ICED™ installation that make it easy to create command files that execute a command string in all suitable subcells of the current cell.

_LOOP.CMD This command file executes a command string in all directly editable subcells of the current cell. Cells that are directly editable are stored in direct-edit libraries. Cells that are stored in protected libraries are not modified.

_LOOP2.CMD Similar to _LOOP.CMD, but with the additional restrictions:

- Subcells at the lowest depth are edited first, then cells at the next level up, etc.
- By changing the value of the loop.level macro, you can also edit subcells in protected libraries. (Cells in view-only libraries cannot be changed. Modified cells from copy-edit libraries will be stored in the current directory, leaving the protected library itself intact.)

Both of these command files use the system macros CELL.NAME.*n*, where *n* is an index integer that is unique for every cell currently loaded into the editor database. The highest valid index is stored in the system macro MAX.CELL.

The MARK_SUBCELLS command is used to create SUBCELL.EDIT.*n* macros that store the edit status of each cell in the database. This allows a command file to determine whether a given cell is stored in a protected cell library.

These command files are intended to be called as helper command files. You can override the values in the default local macros on the command line when you call these command files using the syntax in the example on page 239.

However, we will copy and modify the code in the _LOOP.CMD file to a new file to create a command file that opens, and then flags for saving, every subcell in all direct-edit libraries. This will have the result of saving the environment of the root cell in all directly editable subcells.

Remember that when you change a startup command file, the changes are not reflected in any existing cells. When you change a startup command file after cells are created, it can result in different environment settings in different cell files. Our new command file will solve this problem.

We will add a line to the beginning of our command file to force execution of the current startup command file. Then, when we execute our command file in a main cell of a nested design, it will have the effect of saving the environment of a modified startup command file in all editable subcells of the design. This command file will have the same effect as executing a modified startup command file in each existing cell.

First, we need to copy _LOOP.CMD to a new file, REPLACE_ENV.CMD, in the current directory. Then we will edit the copy. You can accomplish this by typing the following in the second console window.

COPY \ICWIN[37]\AUXIL_LOOP.CMD REPLACE_ENV.CMD
NOTEPAD REPLACE_ENV.CMD

The contents of this file are shown below.

```
! LOOP is intended to be launched with a command like:
!
! @LOOP; #LOOP.OP=@DO_SOMETHING_USEFUL
!
! which will DO_SOMETHING_USEFUL in the current cell and each direct editable
! subcell of the current cell.
!
VIEW OFF;
GLOBAL loop.n = 1;
!
! It's normally a good idea to execute the following commands in each subcell:
!
DEFAULT LOCAL #safe.op = &
  "XSELECT OFF; UNSELECT ALL; UNPROTECT ALL; UNBLANK ALL;"
!
! For test purposes:
!
DEFAULT LOCAL #loop.op = "VIEW ALL;  VIEW ON;  VIEW OFF;  PAUSE 1";

MARK_SUBCELLS
WHILE(%loop.n <= %max.cell){
  $$ subcell.edit.%loop.n=%subcell.edit.%loop.n
  if(%subcell.edit.%loop.n==3){
    EDIT CELL %cell.name.%loop.n;
    %safe.op;
    %loop.op;
    LEAVE;
  }
  #loop.n = {%loop.n + 1};
}
%safe.op ! top level cell
%loop.op ! top level cell
```

Figure 78: Q:\ICWIN[37]\AUXIL_LOOP.CMD

[37] Remember that Q: and \ICWIN are used throughout the manual to represent the drive and directory where you have installed the ICED™ software.

The VIEW OFF command at the start of the file prevents the update of the display after every command. Since this command file requires no user interaction, there is no point to refreshing the display while the file executes. Preventing the redrawing of the display will speed up this command file quite a bit.

We needed to copy this command file rather than call the unmodified file because we must change a line of the existing file. The command file as it stands will not save any cells where only environment settings have changed. You must change the LEAVE command in the loop to an EXIT command.

```
if(%subcell.edit.%loop.n==3){
    EDIT CELL %cell.name.%loop.n;
    %safe.op;
    %loop.op;
    EXIT;
  }
```

Now every directly-editable subcell will be flagged for saving.

The current environment is always saved in every cell file flagged for saving. To insure that the current environment is set based on the current startup command file, we need to add the command that executes the current startup command file. Add the following line to the beginning of the file:

```
@*      !This executes the current startup command file.
```

Our command file will now work, but we can simplify it for our purposes. The following command file statements:

```
%safe.op;
%loop.op;
```

execute the strings stored in each of these macros in each cell. We do not need to execute any commands in each cell except for the EXIT command. So we can eliminate these lines, and the lines that create the default macro definition for each macro.

The REPLACE_ENV command file should now look like the following:

```
VIEW OFF;
@*      !This executes the current startup command file.
GLOBAL loop.n = 1;

MARK_SUBCELLS
WHILE(%loop.n <= %max.cell){
  $$ subcell.edit.%loop.n=%subcell.edit.%loop.n
  if(%subcell.edit.%loop.n==3){
    EDIT CELL %cell.name.%loop.n;
    EXIT;
  }
  #loop.n = {%loop.n + 1};
}
```

Save the file.

To test this command file we need to create a nested design. Execute the following menu options to create 2 subcells, add some geometry on layer 1 to the cells (including the main cell), and then add both subcells to the current cell. At the first prompt for a cell name type SUBCELL1, for the second type SUBCELL2. The exact ADD command executed in each cell is not important as long as you add some sort of component on layer 1 to each cell. Different shapes will make it easier to recognize each subcell.

1:**UseLay → By # → 1:NWEL → BOX**	!Use cursor to define component
2:**EDIT →** (EDIT)**cell → KeyBoard**	!Type SUBCELL1
1:(ADD)**wire**	!Use cursor to define component
1:**FILE → EXIT**	
2:**EDIT →** (EDIT)**cell → KeyBoard**	!Type SUBCELL2
1:(ADD)**poly**	!Use cursor to define component
1:**FILE → EXIT**	
1:(ADD)**cell → SUBCELL1**	
1:(ADD)**cell → SUBCELL2**	

Note the color the components are drawn with and the name of layer 1 shown in the command prompt. Now close the editor and save these cell files.

1:**FILE → EXIT**

Back in the lesson on page 231 you created a copy of the startup command file, Q:\ICWIN\TECH\SAMPLES\MYNEW.CMD. Edit this command file now. You can use the console command:

NOTEPAD \ICWIN\TECH\SAMPLES\MYNEW.CMD

Find the line that defines the name and other settings of layer 1. Change the line to look like the following:

LAYER 1 NAME=**NEWNAME** WIDTH=3.000 SPACE=0.000 **MAGENTA** …

Save the file and exit the text editor. Now we need to open the editor using the modified project batch file that specifies this startup command file. In the console window type the following:

MYICWIN NESTCELL[38]

Press the <Esc> key to allow you time to note that the name of the current startup command file is the file we just modified:

!Option: STARTUP=Q:\ICWIN\TECH\SAMPLES\MYNEW

Press <Enter> to replace these messages with the layout view. Note that the name of the default layer in the command prompt has not changed, nor has the color of the geometry. The startup command file has not yet been executed. A startup command file is executed automatically only when the editor is launched to create a new cell.

Now execute your new command file by typing the following on the layout editor command line.

@REPLACE_ENV

Note that now the name of the default layer has changed and that the shapes drawn on layer 1 are now drawn in magenta. Now exit the editor.

1:**FILE → EXIT**

Press **<Esc>** to leave the exit messages on the screen for a moment. Note that the subcell cell files are saved. This new environment was saved in the subcell files as well.

If you had just executed the new startup command file without executing your REPLACE_ENV.CMD file, only the root cell (i.e. main cell) file would have been saved and the environment of the subcell cell files would have remained unchanged.

[38] If you were editing a different cell name when you created the cell with 2 subcells earlier in this lesson, use that cell name instead of "NESTCELL".

The settings for layer 1 in the subcells would be inconsistent with the new startup command file.

Before you use this command file in a real design, if you want to save all subcell cell files, you must change all cell libraries temporarily to direct edit libraries or cells in protected libraries will remain unaltered. See page 66 to learn how to define the protection status of cell libraries. Be sure to back up your cell libraries before attempting this type of procedure.

Manipulating a List of Components

A useful feature of command files is the ability to save a list of components that can be selected one at a time. The LIST command stores the id numbers of all components that are fully or partially selected in a list. You can then use special SELECT commands to select these components one at a time.

(The LIST command and related features are documented more fully in the Command File Programmers Reference Manual. You can also get more details in the Q:\ICWIN\DOC\LIST.TXT file.)

To test this feature, open the layout editor from a console window with the tutorial directory we have been using as the current directory. (If an open console window already has this directory as the current directory, the CD command line below is not required.)

```
CD \ICWIN\TUTOR\CMDFILES
MYICWIN NESTCELL
```

If the cell has no components, use ADD commands to add several components. Now select all components and store a list of them by typing the commands:

```
SELECT ALL
LIST #mylist
```

Now unselect all components and then select the first component on the list with the commands:

```
UNSELECT ALL
SELECT LIST #MYLIST NEXT
```

You can see that one of the components on the list is selected. Repeat the last command with the 1:**Again** option or the <↑> <**Enter**> combination. Now two components should be selected. Keep repeating the SELECT LIST #MYLIST NEXT command until all components are selected. Repeat the command again and note that when you have passed the end of the list, the history line reports that you have "Passed end of list". This is not an error, indeed this is the only way to realize that you have come to the end of the list, when no component is selected by the SELECT LIST command.

However, repeat the command one last time and you will see that an error is raised. The first time you pass the end of the list, is not an error, but the second time is. This will prevent infinite loops in command files that might otherwise loop endlessly trying and failing to select any new components on the list. The error condition will immediately terminate the command file and return you to layout mode.

A few points on lists:

- Select the components for the list prior to defining the list with the LIST command. (If you want to add all components to the list, be sure to include UNPROTECT ALL and UNBLANK ALL commands before the SELECT ALL command.)
- You can define a default list just as you would a default macro. If the DEFAULT keyword appears immediately after the LIST keyword, then the list will not be redefined if an existing list with the same name already exists.
- Define the scope of a new list with the LOCAL or GLOBAL keyword. If you omit both keywords, a GLOBAL list is created.
- The name of the list comes last, prefixed with the '#' character. List names follow same rules as other macro names, except that they can be no longer than 16 characters.
- You can delete a list with the REMOVE command.
- A list is available only while editing the cell in which it was created. For example, if you create a list while editing a subcell, the list is removed automatically once you return to the parent cell.
- The SELECT LIST command should include one of the keywords: FIRST, NEXT, LAST, or PREVIOUS. FIRST or LAST will reset the list index and then select the first or last component on the list. The NEXT keyword is used to traverse the list from the first component to the last. PREVIOUS allows you to traverse the list in the opposite direction; from last to first. When you do not use any of these keywords, NEXT is assumed.
- Components that no longer exist (i.e. they have been deleted, merged, grouped, etc.) since they were put on list are skipped over by the SELECT

LIST commands. Similarly, components that were protected or blanked after being put on the list are skipped over.

Lists are most useful when you use an UNSELECT ALL command just prior to the SELECT LIST command. This will insure that a single component on the list is selected. You need only check that the value of the n.select macro is non-zero to test that the end of the list has not yet been reached.

Snapping Every Coordinate in a Cell to the Resolution Grid

This lesson will use the LIST and ITEM commands in a command file that snaps all coordinates of all components to the resolution grid.

Create a new command file in the Q:\ICWIN\MYCMDFILES directory. (To open a text editor from a console window, type the following at the console prompt. If your second console window is no longer open, open a new one. If your second console window still has MYCMDFILES as the current directory, you do not need the CD command below.)

```
CD MYCMDFILES
NOTEPAD SNAPIT.CMD
```

This command file can take quite a bit longer than the other command files in this tutorial. It needs to look at every component in the current cell and check that all coordinates are on grid. For a dense cell, this may take a few moments. So we will speed it up some by turning off the view window refresh and minimizing the journal file entries. These two features can speed up command files considerably.

Type the following lines in the text editor:

```
VIEW OFF
LOG LEVEL=BRIEF
```

Now we need to build a list of all components in the cell. Before the LIST command is used, we need to make sure that all components are selected. So we need to unprotect and unblank all components first.

```
UNPROTECT ALL; UNBLANK ALL
SELECT ALL
LIST LOCAL #snap.list
```

Next comes the WHILE loop that will handle one component each time through the loop. The SELECT LIST statement will select one component off of the list at a time. The RETURN command will end the command file when the SELECT LIST command fails to select a component. This means that the end of the list has been reached and every component has been handled.

```
WHILE(1){
        UNSELECT ALL
        SELECT LIST #snap.list NEXT
        IF(%n.select==0) RETURN;
```

Since the loop will end when the RETURN statement is executed, we created the Boolean statement in the WHILE command to loop forever. Use caution when writing WHILE loops in this fashion. Since there are no user interaction commands in this loop, it could loop forever with no chance to interrupt it unless you close the editor entirely with an operating system command. However, since the end of the list will definitely be reached, and the RETURN statement executed, this loop is safe enough. (When debugging your own loops, adding a PAUSE statement would be a good idea to give you a chance to terminate the command file without having to close the editor if your loop does not work the first time the way you expected.)

The next statement uses the ITEM command to save the selected component information in a set of macros with names that all begin with "snap.item". Remember that you must be sure that exactly one component is selected to use the ITEM command. The last statement above in our command file insures this.

```
        ITEM GLOBAL #snap.item
```

Now we declare a few other macros we will use in the loop that checks each coordinate. We will use the snap.item.n.points macro created by the ITEM command to store the number of coordinates in this component. The replace macro is a flag that indicates that the component should be replaced since its coordinates needed to be snapped to be on grid.

```
        LOCAL #i=1
        LOCAL #n=%snap.item.n.points
        LOCAL #replace=0
```

You may wonder about the placement of the macro definition statements in the loop. In other programming languages, this might be an error, or at least a wasteful use of storage. However, in ICED™, this method of macro assignment is just as efficient as defining the macros outside of the loop, then using simple assignment statements in the loop. It saves on typing and will execute just as quickly.

The next loop will investigate each coordinate by first using the ROUND function that snaps a position to the resolution grid. The original position is then compared to the results of the ROUND function to test whether the original position was on grid. If it is not, then the replace macro is set to a Boolean TRUE to indicate that the component needs to be replaced. The position in the macro created by the ITEM command is replaced with the snapped position.

```
                    WHILE(%i <= %n){
                              LOCAL #pos = {ROUND(%snap.item.pos.%i)}
                              IF(%pos!=%snap.item.pos.%i){
                                        #replace=1;
                                        #snap.item.pos.%i = %pos;
                              }
                              #i = {%i+1};
                    }
```

Note the curly braces on the statement that increments the loop counter i. The loop would not execute correctly without them. The curly braces force the addition to be performed instead of only simple string substitution. Note also the curly braces around the call to the ROUND function. If they were not present, the statement would be interpreted as a simple string that should be stored in the pos macro. Instead the curly braces force the evaluation of the statement and the ROUND function is called and the result stored in the pos macro.

Next, we replace the selected component with the component information currently stored in the ITEM macros. The add.*item_name* macro always contains a string that can be interpreted as an ADD command to create a component based on the values in the ITEM macros. The DELETE command deletes the original component since that is still the only selected component.

```
                    IF(%replace){
                              %add.snap.item
                              DELETE
                    }
```

Finally, add the closing curly brace to end the outside WHILE loop.

```
          }
```

No success message at the end of the command file is possible with this command file since it will never proceed past the end of the outer WHILE loop. The RETURN statement will end the command file while it is still executing statements in the loop.

Snapping All Coordinates in All Cells Using _LOOP.CMD

Now that we have a command file that will snap to the resolution grid all coordinates of all components in a given cell, it is easy to create a command file that will perform this operation in all cells of your design.

Create a new command file in the Q:\ICWIN\MYCMDFILES directory. (To open a text editor from a console window, type the following at the console prompt. If your second console window is no longer open, open a new one. If your second console window still has MYCMDFILES as the current directory, you do not need the CD command below.)

```
CD MYCMDFILES
NOTEPAD SNAPALL.CMD
```

Type the following line as the entire contents of the file.

```
@_LOOP.CMD; #loop.op="@SNAPIT LOG=OFF";
```

This command line will execute the _LOOP.CMD command file with the macro loop.op set to “@SNAPIT LOG=OFF”. This will execute the SNAPIT.CMD command file in each of the directly editable subcells of the design. (Cells in protected libraries will not be altered. If you wanted to edit cells in all libraries, you could temporarily change all cell libraries to direct-edit libraries in the project batch file.) **Save the file** and exit the Notepad editor.

To test this command file, you need to add some components that have some vertices not on the resolution grid. Hopefully you are still editing the cell with two nested subcells that we created back on page 258. If not, close the editor and open the main cell from that example, or create and add two subcells to whatever cell you are currently editing. You will need to type the ADD BOX commands, since you cannot use the cursor to define coordinates that are not on the resolution grid. So let’s just type all of the commands. Don’t forget to use the history feature (<↑> key) to make it easier to enter the repeated commands.

```
EDIT  CELL  SUBCELL1
ADD  BOX  5.001  10.999  15.123  20.456
EXIT
EDIT  CELL  SUBCELL2
ADD  BOX  5.001  10.999  15.123  20.456
EXIT
ADD  BOX  5.001  10.999  15.123  20.456
```

(If these cells did not already exist in your design, add both cells to the main cell with 1:(ADD)cell.) The last ADD BOX command adds a component to the main cell. This component will be rounded to lie on the resolution grid as well as the shapes in the subcells.

You can use the 2:(SHOW)screen or 2:(SHOW)@deep to see that these new components were created with the indicated coordinates that are not on the resolution grid of .5. Then type the following command to round all of these new components to the resolution grid:

```
@SNAPALL
```

Now use the 2:(SHOW)screen and 2:(SHOW)@deep options to see that the components have indeed been rounded to the resolution grid.

Conclusion

You can use any of the commands described in the ICED™ Layout Editor Reference Manual in command files. In addition, you can use any of the commands listed below. The page number in the table refers to the page where each command is used in a lesson in this tutorial.

Command	**Use**	**Page**

IF (*boolean_exp*)
ELSEIF (*boolean_exp*)
ELSE
	Conditionally execute statements or blocks of statements	244

ITEM [GLOBAL | LOCAL] #*item_name*	Store component information in macros	252, 262
LIST [DEFAULT] [GLOBAL | LOCAL] … … #*list_name*	Create a list of selected components that can be selected one at a time	260, 262
LOG [ON|OFF] [SCREEN=(ON | OFF]… …[LEVEL=(BRIEF | NORMAL | DEBUG]	Enable or disable logging of commands in journal file and control the level of reporting	226, 262
MARK_SUBCELLS	Initialize contents of SUBCELL.EDIT.*n* macros to store edibility of given cells.	254

(Continued on next page.)

PAUSE	Pause command file	252
PROMPT *prompt_string*	Display message to user on screen in command prompt	253
REMOVE *macro_name*	Delete macros	235
RETURN	End command file immediately	244, 262
VIEW [ON | OFF]	Enable or disable screen updates during command file	253 257
WHILE (*boolean_exp*)	Conditionally execute a block of statements more than once	245
XSELECT [ON|OFF]	Enable/Disable embedded SELECT commands	228

Figure 79:Commands used primarily in command files

When debugging command files, it is helpful to use $comments and PAUSE commands to follow the progress of the command file while it is executing. This also allows you to cancel the file. You can cancel a command file by pressing both mouse buttons any time it is waiting for user input.

Remember that you can also look at the journal file to see exactly what commands were executed during the command file. Adding the command LOG=ON LEVEL=DEBUG to the beginning of the command file will add extra information to the journal file.

Always debug new command files extensively before using them in real design cells. The UNDO command will not undo the effects of any commands executed in command files.

You can add commands to your command files that allow you to undo the effects. However this is too detailed a subject to cover in a general tutorial. See the ED.CMD and UNED.CMD command files for an excellent example of how this is done.

Unless you have added special commands that allow the user to undo the effects of your command files, they may have to terminate the editor without saving the cell file(s) if the command file does not work as expected. Use one of the following methods:

QUIT	If changes were made only to the current cell, this command will close the cell without saving the cell file.
JOURNAL	If changes were made to subcells in your command file and the cells were flagged for saving with the EXIT command in the command file, then use the JOURNAL command to close the layout editor. The layout editor is closed immediately and no cell files are saved.

If other work was done in the editor session before the command file that did the damage, then this work can be recovered with the journal file. See the ICED™ Layout Editor Reference Manual for complete details on this process. After ending a session with the JOURNAL command, be sure to delete all lines in the journal file (*cell_name*.JOU) after the comment that indicates the start of the command file before attempting to reopen the cell. The editor will recognize that the last edit session was closed abnormally and will present you with the option to recover your work. It will use the modified journal file to re-execute all commands executed in the last session up to the point where you called the bad command file.

One interesting aspect of command files that was not explored in this tutorial is the ability to use custom menus. This allows the user to select items from a menu list instead of typing responses. For example, the user could select a layer by clicking the name from a list rather than typing the name or number. You can see an example of this in the command file PLOTNOW.CMD. This command file is executed when you use the menu option 1:**FILE** → (PLOT)**now**. Create custom menus with the MkMenu utility.

There are many other useful command files supplied with the installation. Look at the files in Q:\ICWIN\AUXIL and Q:\ICWIN\TECH\SAMPLES directories for many excellent examples.

To learn more about writing command files, read the Command File Programmer's Reference Manual.

Using the Internal DRC

In this tutorial we will cover some very interesting new features of the ICED™ layout editor that allow you to execute the DRC program without leaving the layout editor. The DRC is an external program purchased separately from IC Editors that performs design rule verification and advanced layer generation operations. The new layout editor features allow you to do operations like the following with commands or menu options in the layout editor:

- perform advanced Boolean operations,
- execute mask operations (similar to the old Calma "win cut" commands),
- bloat or shrink layers, and
- run design rules checking with traditional DRC rule sets without leaving the layout editor. New interactive features make rules check results easy to interpret.

If you prefer to skip this tutorial, you can skip ahead to page 301.

Requirements for Using Internal DRC Operations

This feature is unavailable with the DOS versions of the layout editor or the DRC. You must be using the DRC3-NT version 3.23 or higher. To test that you have the version required, type the following at the prompt of a console window opened with the ICED desktop icon.

DRC3-NT

If the system responds that it cannot find the program, or if the banner displayed indicates a version prior to 3.23, you will not be able to perform the lessons. Contact IC Editors technical support for assistance or to purchase DRC maintenance.

You will also need a fairly recent version of the layout editor. You can open the layout editor executable with no parameters to check the version number. Type at the console prompt:

ICED

If the version is older than 4.63, contact IC Editors for a newer version. Type <Enter> to close the program without loading any cells.

Each time the DRC executes, it verifies that you have the security dongle (key) installed. On some computer systems, the operating system responds slowly to the type of system request used to verify our older security dongles. If you experience unacceptable delays or timeouts during these tutorials you may need to contact IC Editors technical support.

Running the DRC internally uses a lot of system resources. If you are using a computer with less than 128Mbytes of memory, you may find the response time of the commands to be frustratingly slow. If you have a large design, you will require even more memory.

Creating the Tutorial Directory

Create a new working directory for these lessons. You can use any method to create the directory and copy the files. To use DOS commands, open a console window with the ICED desktop icon, then type the following at the console prompt.

```
CD TUTOR
MD IDRC
CD IDRC
COPY \ICWIN[39]\SAMPLES\DRC
```

Loading the DRC Menu

To enable the layout editor to use the internal DRC features, you usually load a special menu. This menu looks just like the regular M1 menu, but it has a fourth top-level menu that includes commands for executing internal DRC operations.

There are three ways to load this menu:

- You can modify your project batch file (used to open the layout editor) to always load this menu by adding a "**MENU=DRC**" parameter to the end of the ICED.EXE command line.
- Another way to always load this menu is by adding a "**MENU=DRC**" command to the end of your startup command file.
- Or you can load the menu explicitly by executing a **MENU=DRC** command once the editor is open.

[39] Remember that \ICWIN is used throughout the manual to represent the directory where you have installed the ICED™ software. Replace this string if required.

We have provided a special ICWIN.BAT batch file in this tutorial directory that uses the first method. This file should have been customized by the installation to use your installation directory name. Open the editor now with this batch file by typing the following in the console window with the new IDRC directory as the current directory:

ICWIN DUMMY

(If the special batch file did not execute correctly, open the layout editor with your usual batch file, then execute the command **@Q:\ICWIN**[40]**\SAMPLES\DRC\NEW** to load the special startup command file referred to in the special batch file.)

Once the editor is open, rotate to the fourth top-level menu by pressing the right mouse button three times. (If you do not see a menu that begins with "DRC-OPs", then type **MENU=DRC** at the layout editor command prompt to load the menu.)

Removing One Layer from Another with Aand-B

For this lesson, you will work on a cell supplied with the sample files. Rotate back to the second top-level menu and edit this cell now with the menu options:

2:**EDIT** → (EDIT)**cell** →**BOOL**

Select the overlapping shapes shown in Figure 80 with:

1:(SELECT)**in**

To see what layers these shapes are on, use the menu options:

2:(SHOW)**@one** → **NEXT** → **NEXT** →**RETURN**

If you did that so fast that you did not see each shape individually selected with its component information reported below the command line, do it again more slowly.

[40] Replace Q:\ICWIN with drive and directory where you have installed the ICED™ software.

Reselect both shapes in Figure 80 with:

1:(SELECT)**in**

Now use the internal DRC to remove all selected HOLE shapes from selected M1 shapes with the menu options:

4:(DRC-OPS)**Aand-B → …**

(LayerA)**M1 → …**

(LayerB)**HOLE →…**

(RESULT LAYER)**M1A →…**

(DELETE INPUT?)**Yes**

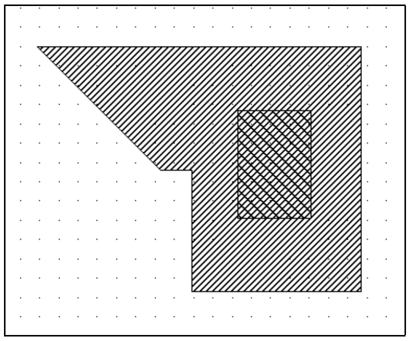


Figure 80: Shapes on HOLE and M1

You will see a slight delay[41] while a command string beginning with "DOS" is shown on the command line. Then the shapes on M1 and HOLE that were selected are replaced with the shapes on M1A shown in Figure 81.

Undoing a DRC Operation

The 1:UNDO menu option cannot undo the results of a DRC operation, but DRC operations can be "undone" using a special menu option. Undo the last operation now with:

4:**@unDRCop**

Redo the effects of the operation now (without re-executing the DRC) by repeating the @unDRCop command with the menu option:

1:**Again**

The 4:@unDRCop option uses the id numbers of the components deleted or created by the internal DRC operation to reverse its effects. If you alter the geometry after the DRC operation, 4:@unDRCop may not work as expected. We will demonstrate this in the following lesson.

[41] The execution time will vary from system to system. The first time you execute a Boolean operation, an extra delay is added while the rule set is created and complied. This delay is avoided when the same operation is repeated. The DRC-NT execution time (reported after the entire operation is complete) should be on the order of a few seconds. If it is significantly longer than this, you may need a new security dongle.

Fixing Cut Lines on DRC Generated Shapes

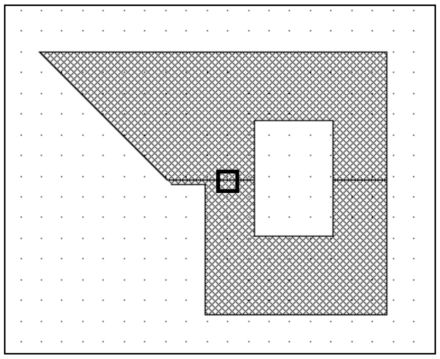


Figure 81: Shapes on M1A

The result of the Aand–B operation is shown in Figure 81. This result shows a side effect of Boolean operations. The result shapes are sometimes cut in places that can cause problems. The long thin segment with the acute angle to the left of the hole may present a problem for some mask-making programs.

This shape was cut because polygons with holes are not valid ICED™ shapes. The polygon with a hole is translated into two valid ICED™ polygons by adding a cut line. Cut lines are occasionally added to translate other types of shapes as well.

A thin slice like this will not be flagged as an error by a DRC (design rules check) program, since shapes on the same layer are merged before any design rules are checked. However, mask-making software may not merge all shapes on a layer before using the data. In this case, a cut line like the one shown in Figure 81 may not cause a design rule violation, but still be unacceptable as valid mask definition.

Most cut lines will cause no problems, but sometimes you must modify the shapes with traditional layout editor commands to get the desired geometry. (The 4:(DRC-OPS)Merge operation cannot be used to fix this problem because the merged DRC shape will still be cut when it is translated back into valid ICED™ polygons.)

The problem shown in Figure 81 is easily fixed with a MERGE command. You cannot remove both cut lines because ICED™ cannot form polygon with a hole. However, you can join the polygons by removing only the cut line on the left.

2:(MERGE)**poly**

Position the near-box for the merge operation as shown in Figure 81. The result is to form one polygon with a cut line only on the right, where it should cause no design rule problems.

Suppose that at this point, you decide that you are not happy with the results and you want to undo the results of the entire Boolean operation. Attempt to perform the reversal command that worked previously.

4:**@unDRCop**

This reversal did not work so smoothly. Now you have shapes on all three layers. This is because you have already deleted the shapes produced by the Aand-B operation and added a new component. The saved id numbers of the shapes affected by the DRC operation are no longer valid and the @unDRCop operation will no longer be able to reverse the operation successfully.

The moral of this lesson is that you should not perform operations that remove or replace components after a DRC operation until you are satisfied that you will not need to undo the DRC operation.

In this case, we can just delete the component on M1A to complete the undo operation.

1:(SELECT)**layer → M1A → Return → ALL**

1:**DELETE**

DRC Operations on Wires

You do not need to select the components to be operated on before performing an internal DRC operation. In this case, the editor automatically executes a SELECT IN operation to select the components for the Boolean operation. Let us demonstrate this with a Boolean XOR operation.

4:(DRC-OPS)**AxorB →** (LayerA)**M1 →** (LayerB)**HOLE →…**

(RESULT LAYER)**M1A**

At this point, the editor executes the SELECT IN command and waits for you to select the components for the operation. Digitize the selection box so that all three shapes on M1 are selected including the wire on the far right.

Once the shapes are selected, you should choose “Yes” to respond to the questions on the next two menus:

(CONVERT WIRES->POLYS?)**Yes →** (DELETE INPUT?)**Yes**

Look at the result of the operation, especially the shape on the far right. Use the SHOW command to get information on this shape.

2:(SHOW)**@one**

You can see that this is a polygon on layer M1A, not a wire. It has the same geometry as the equivalent wire component.

The DRC operates only on polygons, so all wires must be converted to polygons before the operation. The result shapes created are always polygons as well.

Ordinarily, the conversion of wires to equivalent polygons is not a problem. However, **if some wires have segments that are not horizontal or vertical, then the polygon formed by the wire's outline will almost certainly have vertices that are off grid**. The coordinates for the off-grid vertices will be rounded to the nearest data base unit to be as accurate as possible. These shapes with coordinates that are off the resolution grid can cause problems for some mask-making software.

If you do not want the result of the internal DRC operation to transform your wires to polygons, then you must leave them out of the internal DRC operation and perform the equivalent operation on the wires separately with ordinary layout editor commands. When the editor has warned you that wires are selected with the (CONVERT WIRES->POLYS?) prompt, reply with a NO. Then the wires are unselected and the operation continues without them.

Wires are not the only components that are left out of DRC operations. Arrays, cells, lines, and text components are automatically unselected before any internal DRC operation.

Only boxes and polygons (including wires converted to polygons) **in the current cell** are operated on by the operations under the (DRC-OPS) heading or the BOOLOP.CMD command file.

Now unselect the shape so that our next Boolean operation will wait for you to select shapes for the operation rather than executing on only the selected shape.

1:(UNSEL)**all**

Unbounded Boolean Operations

All of the operations on the 4:(DRC-OPS) menu are bounded operations. The boundary of the shapes in the operation determines the boundary of any shapes on the result layer.

An unbounded Boolean operation is one that involves the inverse of a layer in such a way that the true result is a shape that surrounds the entire design and extends to infinity. For example, a simple Boolean NOT operation on a single box results in an unbounded shape that extends to infinity in all directions with a hole in it.

To perform the equivalent operation, the DRC creates an arbitrary boundary at the bounding box of the cell. The bounding box is the smallest rectangle that contains **all** of the geometry in the cell. The bounding box of the cell (and therefore the boundary of the "unbounded" DRC operation) may extend to surround shapes on layers not involved in the DRC operation.

To use an unbounded Boolean operation, you must use the complete Boolean operations menu by executing the BOOLOP.CMD command file. (This special menu is available even when you do not load the DRC.MEN menu.) Type:

@BOOLOP

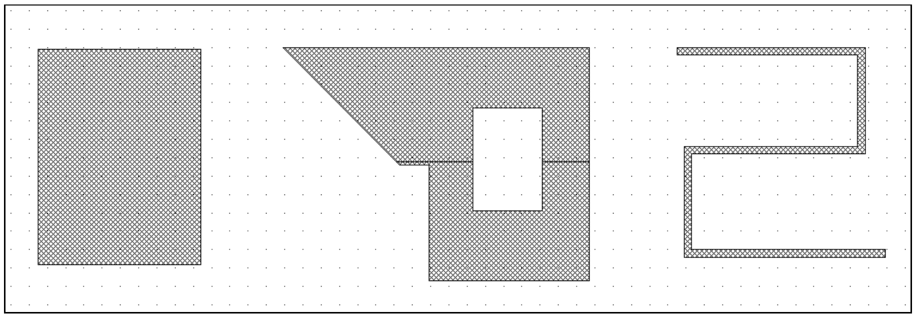


Figure 82: Shapes on M1A

We will indicate selections on this menu with the string “B_OP”. To perform a Boolean NOT operation on shapes on the M1A layer shown in Figure 82, select the following options:

B_OP:(UNBOUNDED OPS)-**A** → (LayerA)**M1A** → (RESULT LAYER)**M1B** → (DELETE INPUT?)**Yes**

When the editor waits for you to select the shapes, select all of the shapes shown in Figure 82.

The result is shown in Figure 83. Note the thin sliver created in the upper left corner of the cell. This is due to the fact that unbounded operations must use the bounding box of the cell as the boundary of the operation. This can lead to unfortunate results like this sliver. Since this sliver is likely to cause minimum width violations, you may need to edit shapes like this with other editor commands. In this case, you could move the end side of the sliver to the right.

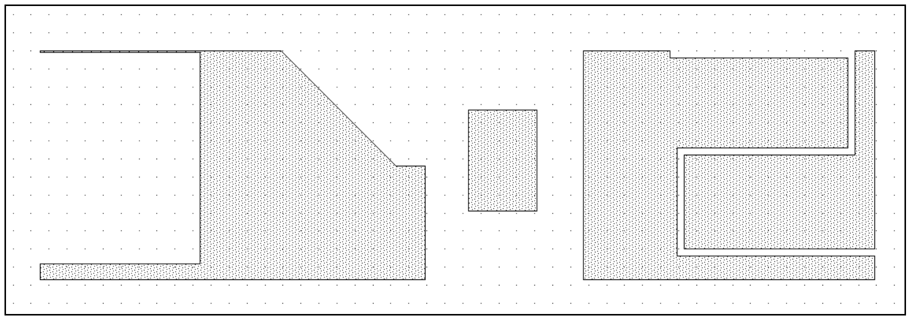


Figure 83: Shapes on M1B = NOT M1A

Another option is to change the bounding box of the cell and repeat the operation. Let us try this method now. First undo the unbounded operation to restore the original geometry shown in Figure 82.

4:**@unDRCop**

Now add a small box on a non-design layer to change the bounding box of the cell.

1:**UseLay → By # → 20 → Box**

Add a small box just **above** the other shapes. You have now changed the bounding box of the cell. Repeat the unbounded operation exactly as before.

@BOOLOP

B_OP:(UNBOUNDED OPS)-**A** → (LayerA)**M1A** → (RESULT LAYER)**M1B** → (DELETE INPUT?)**Yes**

You do not need to select the new shape when selecting shapes for the –A operation. The shape on layer 20 will not be involved in the actual Boolean operation, it merely changes the boundary of the cell used as the boundary of the operation.

Your results should look similar to Figure 84. You can see that the sliver is not created since the boundary of the operation is now well above the other shapes. At this point you could delete the shape on layer 20.

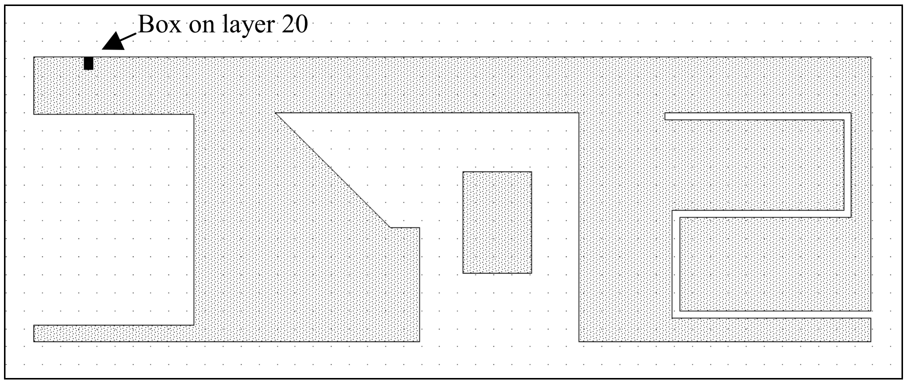


Figure 84: Unbounded NOT M1A results after changing cell boundary

Merging Shapes Using the DRC

Unlike the layout editor's MERGE command, the Internal DRC Merge operation allows shapes to overlap. Furthermore, you do not need to identify the touching edges. The DRC MERGE operation will combine all selected overlapping shapes on the same layer. You do not even need to select a layer for the operation. The original shapes are automatically removed and replaced with the merged shapes on their original layer(s).

This operation can be used to replace self-intersecting wires with valid geometry. (Self-intersecting wires can create problems for some types of mask-processing software.) Create a self-intersecting wire that represents the letter 'P' now with the following menu options. Create the shape shown in Figure 85. Press the right mouse button after digitizing the last point of the wire to indicate that you have finished the component.

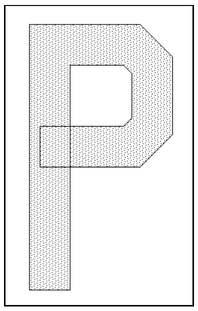


Figure 85: Self-intersecting shape on M1B

1:**UseLay → M1B → Wire**

Select the shape before the DRC operation this time.

1:(SELECT)**new**

4:(DRC-OPS)**Merge →** (CONVERT WIRES->POLYS?)**Yes**

The shapes shown in Figure 86 are the result. Since this operation had resulted in a shape with a hole, the shape was cut to form valid ICED™ polygons. The DRC Merge operation cannot be used to combine shapes with holes into single shapes since shapes with true holes are not valid ICED™ polygons. See page 273 to learn how to use the 2:(MERGE)poly operation to remove only one cut line to form a single valid polygon with a hole.

We are now finished with the BOOL cell. Close this cell without saving any of the changes with the menu options:

1:**FILE → QUIT**

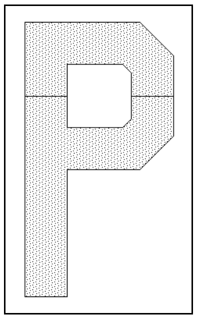


Figure 86: M1B after MERGE operation

Bloating a Layer

This lesson and the next will need the geometry in the cell MASK. Open this cell with the menu options

2:**EDIT** → …

… (EDIT)**cell** → …

… **MASK**

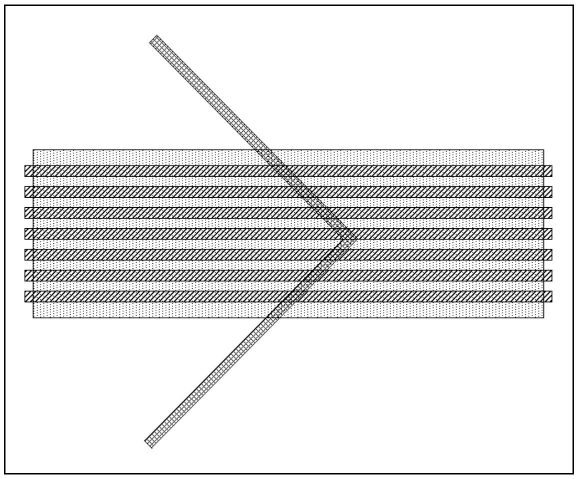


Figure 87: Shapes in MASK cell

The shapes in the cell are shown in Figure 87. In this lesson we will bloat the angled green wire on layer M1A. In a following lesson, we will use this bloated outline as a mask to cut away all shapes near the wire.

4:(DRC-OPS)**Bloat →** (BLOAT/SHRINK LENGTH)**1.000 →** …

… (LayerA)**M1A →** (RESULT LAYER) **MASK**

Select the bent wire at this point and then continue the operation with:

(CONVERT WIRES->POLYS?)**Yes →** (DELETE INPUT)**No**

Note that this time the original shape on M1A is not deleted. In fact it should still be selected. See that is still a wire with:

2:(SHOW)**@one**

The wire was converted to a polygon in the DRC processing, but that internal representation has been discarded. The original wire remains unchanged in the layout editor. The yellow polygon created on layer MASK surrounding the M1A wire is the result of the bloat operation. Select this shape as well now and then display information on both components at the same time with the options:

1:(SELECT)**in**

2:(SHOW)**screen**

Note that the vertices of the wire are on a .5 micron resolution grid while the vertices of the polygon are off grid. This is an unavoidable result of bloating a shape with skewed sides. Press <**Enter**> when are done looking at the results of the SHOW command.

Mask Operations

The mask operations (*andM, *and-M, and WinCut) are the equivalent of the old Calma "Win Cut" commands. While the other Boolean operations act on one or two layers at a time, the mask operations act between selected shapes on a mask layer and all selected shapes on each of the other 254 layers.

You are not given the option of whether or not to delete the input shapes for these operations. The original shapes are deleted automatically, including the mask shapes.

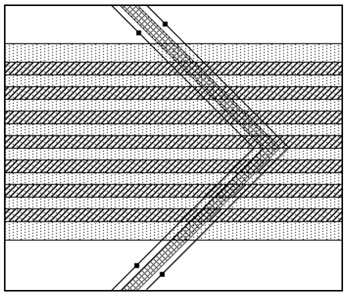


Figure 88: Bloated shape on MASK layer selected

The *andM Operation

This operation will remove from selected shapes on all layers all area not covered by selected shapes on a mask layer.

The rules file generated for a *andM command contains 254 blocks of the form:

INPUT LAYER *i* I*i*;
OUTPUT LAYER *i* X*i*;
X*i* = I*i* AND *mask_layer*;

where *i* is any integer in the range [1:255] except for the *mask_layer* number.

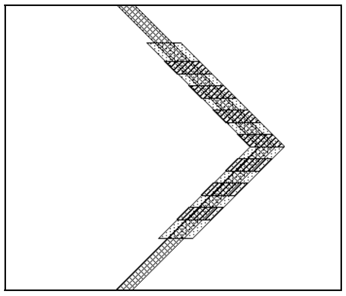


Figure 89: *andM results

The *and-M Operation

This operation will remove from selected shapes on all layers all area that is covered by selected shapes on a mask layer. The rules file generated for a *and-M command contains 254 blocks of the form:

INPUT LAYER *i* I*i*;
OUTPUT LAYER *i* X*i*;
X*i* = I*i* AND NOT mask_layer;

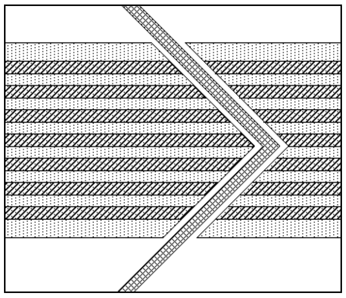


Figure 90: *and-M results

The WinCut Operation

This operation combines the *andM and *and-M operations. All selected shapes on all layers are cut at the boundaries of selected shapes on the mask layer. The mask shapes are deleted along with all of the other selected shapes. The other selected shapes are replaced with polygons cut at the mask shape edges.

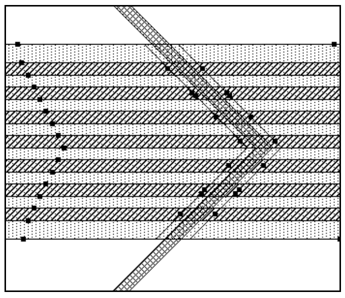


Figure 91: WinCut results with shapes on left and right selected

Masking Around a Wire

In this lesson we will perform a *and-M operation to remove from all shapes the area covered by the bloated shape on layer MASK.

Before we begin the internal DRC operation, we will select the shapes. This is particularly important for our lesson. If the green wire on M1A is selected when we perform the internal DRC operation, then it too will be deleted, since it is covered by the shape on layer MASK.

When no components are selected at the start of an internal DRC operation, a simple SELECT IN command is generated. It can be difficult to select the shapes for a masking operation with this single command. It is much easier to select all shapes on required layers prior to the masking operation. **It is important to select all shapes on the masking layer as well.**

1:(UNSEL)**all**

1:(SELECT)**layer →MASK → M1 → M2 → Return → ALL**

4:(DRC-OPS)***and-M →** (MaskLay) **MASK →** …

… (CONVERT WIRES->POLYS?)**Yes**

It is important to choose Yes at the "CONVERT WIRES->POLYS?" question. Remember that the DRC can process only polygons. If you chose No at the above prompt, then the wires would have been unselected and would not have been masked by the operation.

If you selected the shapes correctly, the results should look similar to Figure 90. The M1 wires are now polygons. The angular cuts have resulted in off-grid vertices for these polygons. This is an unavoidable result of intersecting with a skewed cut line.

Snapping DRC Generated Shapes to Grid

If you did the tutorial on command files, you created a SNAPIT.CMD command file (see page 262) that will snap all of the coordinates in this cell to grid. If you did this lesson, let us execute this command file now to snap all of the coordinates to grid. If you have not already done this lesson, skip ahead to the next lesson.

To execute this command file, you will need to specify the path to the file. The batch file used to open the layout editor did not add this directory to the command file search path. Type:

@Q:\ICWIN[42]\MYCMDFILES\SNAPIT

Let us look closely at the new shapes on the M1 layer verses the old off grid shapes. Figure 92 shows both the old and new M1 layers near the center of the cell.

In this case, the snap went quite well, no shapes were significantly distorted and all coordinates were shifted away from the wire rather than closer to it. Not all snapping operations will go so smoothly, but it may be a good place to start if you need all coordinates to be on grid.

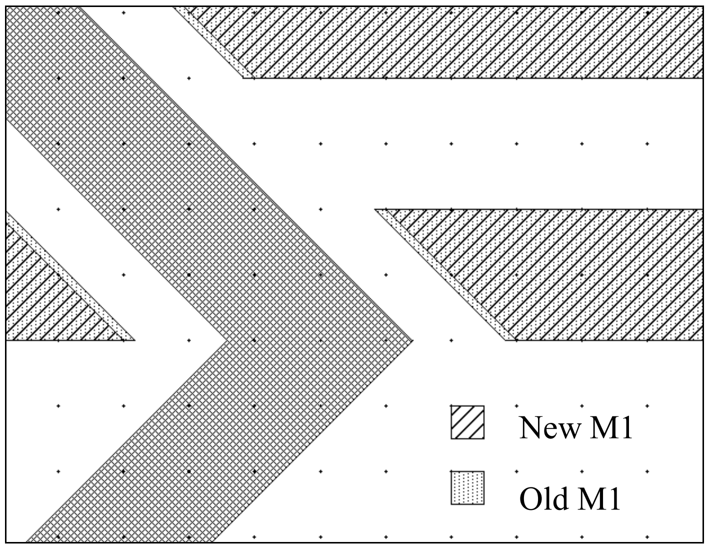


Figure 92: M1 shapes before and after snapping to grid

[42] Replace Q:\ICWIN with drive and directory where you have installed the ICED™ software.

Performing DRC Design Rules Checks Internally

You can use the DRC for other processing than simple Boolean operations. If you already use the DRC separately from the layout editor with a rule set designed for your technology, you can now check the design rules without leaving the layout editor session. This is best done on relatively small cells, since the DRC will run more efficiently outside of the layout editor with all of your system's resources.

If you have not yet used the DRC for design rules checking, this is a good time to start. The new interactive features make it easier than ever to find and fix design errors with the DRC.

We will use a new cell for this lesson. Close the previous cell without saving it, and then close the editor.

1:**FILE → QUIT**

1:**FILE → QUIT**

Before we open the layout editor again, we need to change the value of an environment variable used by the internal DRC. The value stored in the **DRC_PATH** environment variable is used as the search path for rules files. **A customized rules file for DRC processing must be stored in a directory on this path to be used by the internal DRC.** This value is set in the batch file used to open the layout editor.

We will edit the batch file now to change this search path. (If you were unable to use the copy of ICWIN.BAT in the tutorial directory, either fix this batch file now, or make this change to a copy of the batch file you use, and then use that batch file to open the layout editor below.) In the console window, the Q:\ICWIN\TUTOR\IDRC directory should still be the current directory. Type the following:

NOTEPAD ICWIN.BAT

Find the line that sets DRC_PATH and change the directory to our tutorial directory.

Old: set drc_path=q:\icwin\tutor;

New: set drc_path=q:\icwin\tutor**\idrc**;

Now **save the file** and close the Notepad editor. Then open the layout editor again by typing in the console window:

ICWIN DUMMY

Add the existing cell TRIVIAL to our empty cell and change the view window to display it well.

1:(ADD)**cell → TRIVIAL**

1:(VIEW)**all**

The contents of this cell are shown in Figure 93. The layer names used in our sample rules file do not agree with the layer names stored in the cell file, so we will ignore the layer names used in the layout editor. We will refer to layers by the layer names used in the sample rules file; layer BOXES for layer number 1 and layer ONE_BOX for layer number 3.

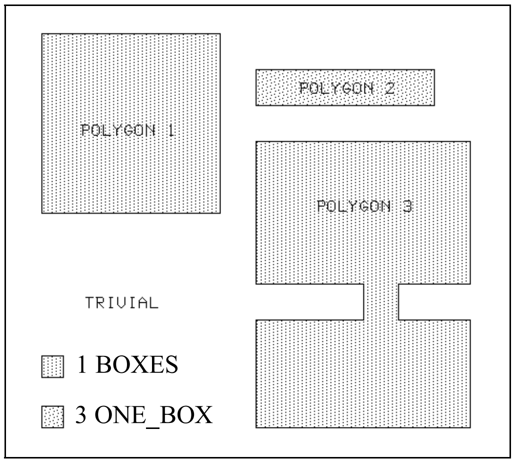


Figure 93: TRIVIAL cell

Compiling Rules Files

The rules we will verify on this cell are shown in Figure 94. (A few other lines are included that are beyond the scope of this tutorial. You should read the DRC manual if you will be writing your own rule sets.)

```
INPUT  LAYER 1 boxes; 3 one_box;
OUTPUT ERROR LAYER 20 too_close1; 21 too_close2;
OUTPUT ERROR LAYER 20 narrow;

CONST min_distance=10;
too_close1=MINSPACING(boxes, boxes, min_distance)
too_close2=MINSPACING(boxes, one_box, 7)
narrow=MINWIDTH(boxes, min_distance);
```

Figure 94: Relevant contents of EX2.RUL

You must compile the rules file as a separate step before executing the DRC to verify the rules in the file. There is no menu option for this procedure, but it only needs to be done once, and then the compiled rule set can be used over and over.

(If you have been using an older version of the DRC and using a rule set compiled with D3RULES.EXE, you will need to recompile the rules set for this newer version of the DRC. The DRC will warn you that the rules set needs to be recompiled and will provide you with the option to do this automatically.)

The rules compiler command line must be typed at a DOS console prompt. An easy way to do this is to open a new window with a SPAWN command typed in the layout editor. Type the following in the layout editor window:

SPAWN

Then at the console prompt in the new window type the following to compile the rules in the EX2.RUL file in the current directory:

D3RUL-NT EX2

If there were syntax errors in the rules set, the compiler would post error messages. You would need to fix the errors and re-execute the compiler. However, since our sample rules set has no errors, the compiler messages end with the word “Done”.

You can leave this console window open and return to the layout editor with the Windows task bar (usually on the very bottom of your screen) or simply by clicking the mouse in the layout editor window anytime it is displayed on the screen. You can return to the console window later with the task bar or by clicking in the window with the mouse.

When you are done with the console window, you can close it with the 'X' button on the upper right hand corner, or by typing "EXIT" at the console prompt.

The DRCSTEP Menu

There is an additional menu that makes running your own rule sets and interpreting the results easier. Load this menu now with the option:

4:**DRCSTEP**

This new menu is added to the front of the top-level menus. The other top-level menus are still available. We will refer to options on this new menu with the top-level menu indicator “DRCSTEP:” This special menu will remain in loaded in the front of the top-level menu list until you

- exit the cell,
- use the DRCSTEP:**LAYOUT** menu option to remove the special menu, or
- use the DRCSTEP:**DRCTAG** or 4:**DRCTAG** options to replace this menu with the other special menu we will discuss in a later lesson.

From the DRCSTEP menu you can execute the DRC using your own rule sets.

Using DRCNOW to Run the DRC

To execute the design rule checks in the compiled file EX2.BB on all the shapes in the current cell (and its subcells), use the menu options:

DRCSTEP:**@DRCNOW** → **EX2** → (OPTIONS)**none**

Choosing (OPTIONS)**some** instead would have allowed you to select a subset of the current design for the rules check. You can choose either to test all shapes inside of a selection rectangle, or to test only selected shapes. The second option is most useful when you select the shapes prior to beginning the internal DRC operation.

Using (OPTIONS)**all** is useful only if you are familiar with the command line options of the DRC. Use this option only when you know you need to override a default on the DRC command line.

When you choose (OPTIONS)**none** and therefore use the default POK mode (or explicitly choose (POK MODE)all with the (OPTIONS)some or (OPTIONS)all choices), all shapes in the cell including shapes in subcells are exported to the DRC. This is different than the default behavior for the Boolean operations.

The default for @DRCNOW design checks is to export all shapes in the current cell and its subcells. The DRC will flatten the design and verify all rules on the entire layout.

When the DRC operation completes, the raw error count and the run time are shown on the history line of the editor window. The raw error count is the number of shapes created on DRC error layers, rather than a count of violations. A single design rules violation can result in several error shapes.

The error shapes are imported automatically (i.e. the DUMMY.NOW command file is executed) and displayed in the editor. They should look like those shown in Figure 95.

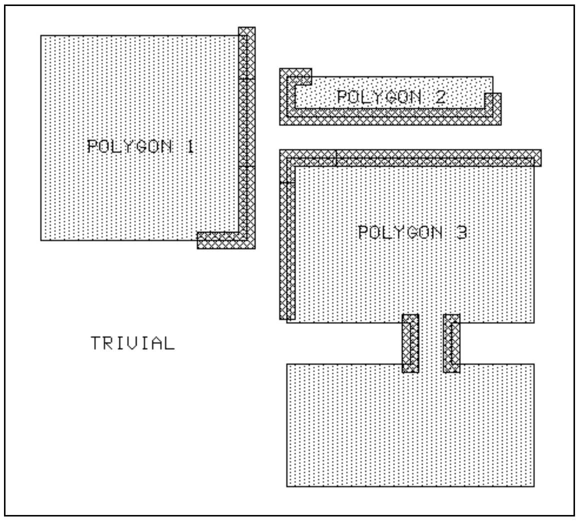


Figure 95: Violations marked with error wires

These shapes are created in the current cell, even though the design errors are in the nested cell TRIVIAL. All shapes created by the DRC are created at the level of the current cell. (The only exception is when you use the HIERARCHICAL option on the DRC command line to create non-error shapes hierarchically. You must read about this option in the DRC Reference Manual to use this feature.) Since we opened the layout editor to edit the non-design cell DUMMY, this is the only cell modified by the internal DRC operation. The cell TRIVIAL has not been modified at all.

To fix the design errors marked by the shapes created in the DUMMY cell, you would need to execute a nested edit of the TRIVIAL cell. The P_EDIT NEAR command is very useful in this situation. However, we will not be fixing design errors in this tutorial.

The use of a dummy cell is not necessary for using the internal DRC for design checks. You can execute the internal DRC directly from a design cell and the error marks (and other output shapes) will be created directly in the design cell. This makes it easier to fix the errors indicated; however the error marks are stored in the cell definition. You will almost certainly want to remove them again from your cell at some point. We will cover new internal DRC options on the DRCSTEP menu that make this task easy beginning on page 293.

Stepping Through Errors One Rule at a Time

The options on the DRCSTEP menu are designed to allow you to step though all of the errors found in the order of the rules that were violated. The (DRC STEP)init option places all of the error marks in a list, then the first, next, prev, and last options under the same heading allow you to travel through the list. The list is preserved even while other components are selected and manipulated.

To demonstrate this feature, select the following menu options to create a list of all error marks:

DRCSTEP:**UNSELECT→ ALL**

DRCSTEP:(DRC STEP)**init**

The fist error mark on the list is selected automatically. The history line reports the following to tell you why this error mark was created.

```
$(1/7) TAG=3:TOO_CLOSE1
```

The “(1/7)” entry refers to the fact that error number 1 of a list of 7 error marks is selected. We will discuss the concept of error tags in a moment, so ignore the “TAG=3” reference for now. The “TOO_CLOSE1” refers to the layer name in the DRC rules set for this error. Look at the listing of the rules on page 285. You can see that TOO_CLOSE1 is the layer used to mark minimum spacing errors between shapes on layer BOXES. This first error mark indicates where the edges of polygon 1 are too close to edges of polygon 3. To select the next error mark, the corresponding error mark on polygon3, select the option:

DRCSTEP:(DRC STEP)**next**

At this point, you can use the options on any of the menus to investigate and then fix the error. To move on to the next violation, execute the next option again:

DRCSTEP:(DRC STEP)**next**

Error mark 3 of 7 is now selected. Note that the history line reports the layer of this error mark as TOO_CLOSE2. This error marks the violation where the edge of polygon 1 is too close to polygon 2.

If you don’t have a listing of the rules in front of you, you may forget what design rule results in a violation on a specific error layer. You can open a Notepad window loaded with the log file from the rules compiler that includes a listing of the rule set with the following menu option. Do this now.

DRCSTEP:**edit.RLO**

Note that TOO_CLOSE2 is the layer for minimum space violations between BOXES and ONE_BOX. Leave this Notepad window open for now. We will look at its contents more closely in a following lesson.

Move on to the next error mark in the layout editor with:

DRCSTEP:(DRC STEP)**next**

Now use the Again option below several times until you come to the end of the list:

DRCSTEP:**Again**

The list created by (DRC STEP)init is lost when the layout editor is closed. If you have not yet come to the end of a list of error marks when you need to close a cell at the end of the day, you would need to recreate the list with (DRC STEP)init in a new layout editor session the next day. In this case the entire list is recreated and your place in the middle is not recorded. You would need to keep executing the next option until you get back to the place where you left off fixing errors. Alternately, you can delete the error marks as you fix each error. In this case, the deleted error marks will not be put on the new list when you reopen the cell and use (DRC STEP)init at a later time. (We explore ways to delete the error marks in a later lesson.)

Changing the View Window to See Each Error Mark

So far, you have not needed to change the view window to see every error mark. However, in most real cells, you will probably want to keep the magnification of the view window at a relatively fine scale to fix each error. When the view window is at a fine scale and you use the DRCSTEP:(DRC STEP)next option, the next selected error shape is probably not displayed in the current view window. Let us define a key macro to change the view window to see each selected error with a single keystroke.

First change the view scale to show only a small part of our sample cell. Then select the first error mark.

DRCSTEP:(VIEW)**in %→ 3.0**

DRCSTEP:(DRC STEP)**first**

The selected error mark is probably not displayed. To change the view window to show the selected error mark and the surrounding geometry, we will assign a view command to the key combination <Shift><F8>. Type the following:

KEY SF8 "VIEW SEL OUT 2"

Now hold the <Shift> key down while you press the <F8> key. The selected error is now displayed and the view window is zoomed out to show enough geometry to show why the error is marked.

Now move on to the next error mark:

DRCSTEP:(DRC STEP)**next**

Change the view window with by using <Shift><F8>.

You could add the commands generated by the DRCSTEP:(DRC STEP)next menu option to the key macro. In this case, you can move on to the next error and display it in the view window with the single keystroke. (To determine the exact syntax of commands generated by menu options, view the journal file with the

1:FILE→edit.JOU option.) Redefine the commands assigned to the <Shift><F8> key combination by typing the following command all on one line:

KEY SF8 "UNSELECT ALL; …

… SELECT LIST=STEP.ERROR.LIST NEXT; VIEW SEL OUT 2"

Now when you type <Shift><F8>, you move on to the next error mark without having to select the menu option, and the error is displayed, all in one operation. If you like using such key shortcuts, you should add key definitions like the one above to a command file that you can execute as desired, or to your startup command file that will be executed in all new cells.

DRC Error Tags

All shapes created by DRC error rules are created with a tag number. (Ordinarily, these are the only shapes with non-zero tag numbers. If you have very specialized creation methods that use undocumented features of ICED™, you may want to make sure that you have not assigned tag numbers to ordinary shapes.)

The tag numbers are related to individual rules in the rule set. The tag number assigned to a DRC error shape will be the number of the rule as reported in the rules compiler log file. If the Notepad window displaying the contents of the rules compiler log file is not still open, open another now with the option:

DRCSTEP:**edit.RLO**

Note the lines near the end of the file that report the rules executed in each pass. In this case, all three of the design error rules are executed in the third pass.

```
Pass 3:
  3. TOO_CLOSE1[20] = MIN_SPACING(BOXES[1], BOXES[1], 10
          +~CONN/P/OVER/CROSS/T/END/~DET)
          (Rules line 8)
  4. TOO_CLOSE2[21] = MIN_SPACING(BOXES[1], ONE_BOX[3], 7
          /+~CONN/P/OVER/CROSS/T/END/~DET)
          (Rules line 9)
  5. NARROW[20] = MIN_WIDTH(BOXES[1], 10/~DET)
          (Rules line 10)
```

Note that shapes on both error layers TOO_CLOSE1 and NARROW are stored on layer number 20. However, these layers are kept separate during the DRC run. The error shapes generated by several different rules may all be on same layer, but they will have different error tag numbers.

Keep in mind that the DRC layer names are not used by the layout editor, except by the new features that relate tag numbers to DRC layer names. Layer number 20 can have any name in the layout editor. However, shapes on this layer created by the DRC will have tag numbers of 3 or 5, depending on which rule created the error shape.

Looking up rule numbers in the compiler log file is somewhat cumbersome, so ICED™ now supports features that automatically relate the tag number to the name of the DRC layer created by the corresponding rule. This information is saved in macros created by the *rule_name.*TAG command file. This command file is executed automatically by any of the features that need this information, so you should never need to execute this file, or even look at it.

These tag numbers are used by several features, so we have added a new menu with options that deal specifically with tag numbers. Switch to this menu now with:

DRCSTEP:**DRCTAG**

The new menu replaces the DRCSTEP menu. You could reload the DRCSTEP menu by choosing DRCTAG:DRCSTEP, but don't do this now. Most of the options on the DRCSTEP menu are repeated here, however the (DRC STEP) options to build and control a list of error shapes are replaced with three options that involve error tags:

LoadMACs
LoadTAGs, and
@DRCTAG

The LoadMACs and LoadTAGs options are used to reload the macros that store DRC information, including tag definitions. Normally, you should never need to execute these options yourself. While the information stored in macros is discarded when you exit a cell, the DRC macro information is normally reloaded automatically by options that use it.

The @DRCTAG option is more useful. It is used to report the tag number and error layer name for specific shapes. It operates in a similar manner to the 2:(SHOW)@ONE option.

To demonstrate this option, change the view window to see the entire layout and then use the @DRCTAG option to display the DRC information on the selected error(s).

DRCTAG:(VIEW)**all**

DRCTAG:@**DRCTAG**

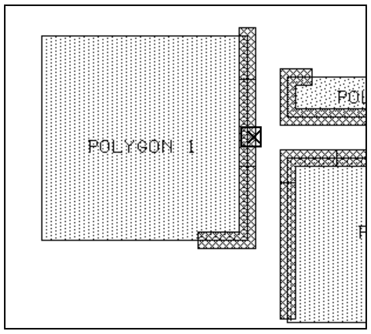


Figure 96: Selecting overlapping error marks

The DRCTAG.CMD command file is now executing. Use the near box to select the overlapping error marks as shown in Figure 96. Only error marks (shapes with non-zero tags) can be selected by this option. In our case, two error marks are selected, so they are put on a list and the first mark on the list is selected and its error information is shown on the history line near the bottom of the window.

```
$ TAG=3:TOO_CLOSE1;
```

This error wire was generated because the edge of polygon 1 is too close to polygon 3. It was created by rule number 3 in the DRC rules compiler log.

```
3. TOO_CLOSE1[20]=MIN_SPACING(BOXES[1],BOXES[1],10)
```

To see the next selected error mark, use the **NEXT** option. When you are finished, use **RETURN**.

Removing DRC Generated Shapes

DRC generated shapes are just like any other shapes aside from the fact that error shapes have non-zero tag numbers. You can delete them just as you would any component. However, it is usually more convenient to delete them with the special options on the DRCSTEP and DRCTAG menus.

Using Unload and Reload

The list of the shapes created by the last run of the DRC is saved in macros. This allows you to easily delete the results, either temporarily, or when you are finished with them. The entire results of the run can be recreated as well, without re-executing the DRC.

To remove all shapes created by the last run of the internal DRC, use the option:

DRCTAG:(@DRCNOW)**unload**

You can restore the shapes now, without re-executing the DRC, with the option:

DRCTAG:(@DRCNOW)**reload**

Reloading twice in row does not add 2 copies of each DRC shape. You can use the error marks for a while, delete some of them, and then use the reload option to restore all error marks from the last DRC run. Any shapes that were loaded from the last run are removed before the entire set is reloaded.

Now execute the DRC again with the same options as above:

DRCTAG: **@DRCNOW → EX2 →** (OPTIONS)**none**

Now try to remove the error marks again with:

DRCTAG:(@DRCNOW)**unload**

It looks as though nothing happened. The error marks are still there. This is because the results from your first run of the DRC are still there. The unload operation only removed the set of shapes created by the last run of the DRC.

Deleting All Shapes on DRC Layers

To delete all of the DRC shapes that may be left behind from previous DRC runs, you must select and then delete them. To make this easier, there are options to select and/or delete shapes by error tags, or by DRC layer numbers.

When your DRC shapes are in a separate cell, as the shapes in our lesson are, you can delete all DRC shapes with confidence. However, if you prefer to run the DRC right from your design cell, then design shapes and DRC shapes are all stored in the same cell and care must be taken to not accidentally delete design shapes along with the DRC shapes. We will emulate this situation by ungrouping the design cell so that all shapes are in the same cell.

2: **EDIT → UnGRP**

Select the outline of the cell TRIVIAL to ungroup it. Now all shapes are stored in the current cell DUMMY.

To select all shapes on all output layer numbers used by the DRC (**this includes non-error layers**), use the option:

DRCTAG:**DRC SEL →** (DRC LAYER TYPE)**both → ALL**

Before you delete these components, note what is selected. Not only are the error marks selected, but the text components are selected as well. The text components are stored on layer number 99 which is used as a default error layer for all DRC runs. (You can override the use of layer 99 with special options. Refer to “Layer number 99” in the index of the DRC manual.)

To safely remove only DRC shapes, you can inspect the remaining shapes before deleting them. You can use the following option to inspect and then delete only the appropriate shapes:

2:(SHOW)**@one**

The editor will now loop through all of the selected components one at a time.

- When a design component is selected, click on **NEXT**.
- When a DRC generated error shape is selected, click on **DELETE → NEXT**.

At first it may seem that the shapes are not being deleted, but that is because there are now two copies of each error shape. Keep deleting components until the DRC shapes are all gone. When the option NEXT is no longer listed, you have come to the end of the list. Return from the 2:(SHOW)@one operation with **RETURN**.

Even with this method of deleting all DRC shapes, you can restore the DRC shapes from the last DRC run with the reload option. You could even exit the cell now, open it again tomorrow, reload the DRCSTEP menu and reload the DRC results. Reload the DRC results now without exiting the editor.

DRCTAG:(@DRCNOW)**reload**

To avoid selecting design shapes with the DRCSTEP:DRC SEL options, you can restrict the selection to shapes with non-zero error tags. This is not completely foolproof depending on how your DRC rule set is constructed, but it should successfully select only DRC error shapes in ordinary situations.

DRCSTEP:**DRC SEL →** (DRC LAYER TYPE)**error → ALL**

Now only the DRC shapes are selected, you could delete them all with an ordinary DELETE command if desired. However, let us keep them for now.

DRCSTEP:**UNSELECT → ALL**

Deleting Individual DRC Error Shapes Safely

If you prefer to delete individual error marks as you fix each error, you can use the ordinary 1:DELETE option. However, if your error mark is difficult to select individually because it is on top of design geometry, you can use the following option that deletes only shapes with a non-zero tag number.

DRCTAG:**ERR DEL**

Repeat this operation a few times with the DRCTAG:Again option. At this point, you could exit the cell, reopen it again tomorrow, and resume fixing errors where you left off with DRCSTEP:(DRC STEP)init.

Writing Rules Sets for Internal DRC Processing

You can use the same rule sets for internal DRC processing that you used when executing the DRC separately. However, when you are writing new rule sets, you should keep the following recommendations in mind:

- Use meaningful layer names in the rule set for DRC error layers. Using layer names like ERR_SPACE_M1_M1 or M1_SPACE_ERRORS will make it easier to diagnose a violation while using the DCRSTEP or DRCTAG options than generic layer names like TOO_CLOSE or ERR_LAY.
- Define all output layers with the ERROR keyword. This insures that all shapes generated by the DRC are assigned tags and can be easily identified and removed.
- If your design uses layer number 99 as a design layer, add BADPOLY and WARN_ACUTE (or NO_WARN_ACUTE) rules to your rule set to prevent layer 99 from being used as an error layer.
- If you use the DRC to generate output layers that you do want to be left in the cell (e.g. a generated PSEL layer,) create a separate rule set for this processing from the set used to mark errors.

Looking at DRC Log Files and Macros

As we have mentioned before, if you need to look at the rules compiler log file, you can use the option edit.RLO on either the DRCTAG or DRCSTEP menus. There is also a menu option to look at the DRC log file. This is particularly important if the

DRC did not execute properly for some reason. Look at the log file from the last DRC run with:

DRCTAG:**edit.DLO**

Note that the log includes a count of how many shapes were created on each DRC output layer. The statistics on how long the DRC took to execute are included as well. If the DRC had not executed properly for some reason, this log file would contain error messages describing the problem.

Some of these results are stored in macros for easy reference. Look at the log macros created by the last run of the DRC with:

DRCTAG:**show.log**

To see the entire list of macros created by the internal DRC operations you must type the following SHOW command:

SHOW USER DRC*

Executing the DRC and DRCSTEP without Special Menus

You can use all of the features just described without loading special menus. In fact, you can even use DRCSTEP to investigate the DRC errors one by one when you execute the DRC outside of the layout editor.

To test this, unload the existing error shapes and then close the editor to remove all macros create by the internal DRC operations.

DRCTAG:(@DRCNOW)**unload**

DRCTAG:**FILE → EXIT**

Now that you are back at the console window, execute the DRC as you would have before the internal DRC features were added by typing:

DRC3-NT EX2 DUMMY DRCOUT SLOW

This places the DRC results in the command file DRCOUT.CMD.

Open the layout editor:

ICWIN DUMMY

Unload the DRC menu by typing:

MENU M1

Note that the menus are now the same old menus that you were used to. Now execute the DRC output command file just as you would have before the new features were available. Type:

@DRCOUT

Now the DRC shapes are created in the layout. You can now execute the DRCSTEP.CMD command file. However, the first call to DRCSTEP must include an argument to initialize the list. Type the following:

@DRCSTEP; local #op=init

Now the first error mark is selected and its error layer information is reported on the history line. To move on to the next error mark, type:

@DRCSTEP

Now you can use the 1:Again option to view the rest of the error marks.

You can also use the DRCTAG.CMD file without the menus. Type the following:

UNSEL ALL

@DRCTAG

Click on an error mark to see its error layer information.

Even the option to execute the DRC from within the layout editor is available without the menus. Type the following:

@DRCNOW

You can see that you are now performing the same procedure as the one you performed when you used DRCSTEP:@DRCNOW. Cancel the command file by pressing both mouse buttons.

If you prefer to build your own command file to execute the DRC, you may want to copy the DRCNOW.CMD command file and tailor it for your own uses. You could include a call to DRCSTEP.CMD to initialize the error mark list if you prefer.

To remove the DRC results entirely, you can use the DRCUNLOAD.CMD command file.

@DRCUNLOAD

Executing Boolean Operations without the Special Menu

The Boolean operations can also be used without the new DRC menu. Simply execute one of the following command files.

BOOLOP.CMD	Bounded and unbounded simple Boolean operations
MASKOP.CMD	*and-M , *andM, and WinCut masking operations
SHRINKOP.CMD	Shrink selected shapes on a layer
BLOATOP.CMD	Bloat selected shapes on a layer
UNDRCOP.CMD	Undo last DRC operation and restore original layout

All of the command files mentioned above (including the DRCSTEP.CMD, DRCNOW.CMD and related command files) are stored in the Q:\ICWIN[43]\AUXIL directory. Read the comments in the files to learn even more about DRC related operations in the layout editor.

The first time one of the Boolean DRC operations are executed (either from the DRC menu or directly from the command files above), the appropriate rules file is created, stored, and compiled in the Q:\ICWIN\TMP directory. If you repeat an operation, the existing rules file is reused. This means that repeated Boolean operations will execute faster than the first operation.

Conclusion

This is a relatively new feature to ICED™. Integrating design rules verification with a layout editor has been problematic for other software vendors. The integration works smoothly until you need to do something slightly different than ordinary, and then the tools can work against you, rather than save you time.

We want our integration to be useful to all of our users, so we appreciate feedback and we do take requests. Tell us how the tools could be made more useful for your application.

[43] Remember that Q: and \ICWIN are used throughout the manual to represent the drive and directory where you have installed the ICED™ software.

Recovering from Mistakes or Crashes

This tutorial focuses on how to restore cell files to what they were before a system crash or editing mistake. Most of these recovery methods depend on the journal file that records every command executed in the current layout editor session.

We need a customized project batch file for these lessons. We need to copy the ICWIN.BAT file and change a line. To do this from the console window, open a console window with the ICED desktop icon then type:

```
COPY  ICWIN.BAT  ICRTEST.BAT
NOTEPAD ICRTEST.BAT
```

Find the line that sets the cell library search path and change the protection status of the only cell library on this path by adding a /c switch after the directory name. This will make the directory a copy-edit library. You can now edit cells in this library and the modified versions will be saved in the same directory as the root cell.

```
Old:   set iced_path=q:\icwin\samples;
New:   set iced_path=q:\icwin\samples/c;
```

Save the file and exit the Notepad editor.

Now create a new working directory for these lessons. To use DOS commands, type the following at the console prompt opened with the ICED desktop icon.

```
CD TUTOR
MD RECOVERY
CD RECOVERY
```

Once RECOVERY is the current directory in the console window, open the editor to create a new cell called RECROOT by typing:

```
ICRTEST RECROOT
```

Now add two cells from the Q:\ICWIN\SAMPLES directory:

1:(ADD)**cell → NAND**

1:(ADD)**cell → NextPATH → VIA**

How Cell Files are Saved

Cell files are saved only when the layout editor terminates. Until you terminate the editor, changes are made only to the images of cells loaded into the program's memory. There is no "Save" option.

When you are editing the root cell, you have four options on how to close the layout editor. All of these options are located on the 1:FILE submenu.

EXIT	The root cell file is saved and the editor terminates.
QUIT	All changes to the root cell are lost and the root cell file is not overwritten.
LEAVE	If changes were made to the geometry of the root cell, then the root cell file is saved. If not, then the cell file is not overwritten. This makes this command the most common termination command.
JOURNAL	No changes are saved and no cell files are overwritten.

If you are in a nested edit session (i.e. you have used the EDIT, PEDIT, or TEDIT commands to edit a subcell), you must first use one of the first three options to end the nested session. If you use EXIT or LEAVE to return from the nested edit, the modified cell is flagged for saving. Once you are at the level of the root cell, the EXIT, QUIT, or LEAVE commands will terminate the layout editor as described above. All cells flagged for saving are saved at this time.

You can use the JOURNAL command at any point (including within a nested edit session) and the editor will close immediately without overwriting any cell files.

The complete series of steps for saving each cell file is listed on the next page. If one of these steps fails due to some system problem like a lack of disk space, the layout editor issues a "SAVE ERROR" message, skips the remaining steps for that cell, and goes on to the next cell. (The error messages are displayed on the screen and recorded in the journal file.) It is rare for problems to occur during these steps, but if you get a warning message at this stage:

Do not try to recover using the journal file and automatic recovery. Do not launch the layout editor until you have copied all files to a backup directory and renamed the affected files.

Instead, you should complete the failed/skipped steps by hand. No data has been lost, but rash actions at this point can lose data.

When you exit the layout editor, it goes through the following steps for each cell flagged for saving. Note that the previous version of each cell file is retained, but renamed with a .CL1 file extension. We'll explore these cell backups more in the next lesson.

1. The cell file is written to the file *cell_name*.TMP.
2. If both cell files *cell_name*.CEL and *cell_name*.CL1 already exist, *cell_name*.CL1 is deleted.
3. If the file *cell_name*.CEL exists, it is renamed *cell_name*.CL1 .
4. The file *cell_name*.TMP is renamed *cell_name*.CEL.

After these steps have been completed for all cells, ICED™ renames the session's journal file from *root_cell_name*.JOU to *root_cell_name*.LOG.

Cell Backup Files

One method of cell recovery is to use cell backup files. Each time a cell file is about to be overwritten, the original version is copied to the file *cell_name*.CL1. You can rename a cell backup file to give it a .CEL extension and use it like any other cell file.

Close the editor now and save the cell file:

1:**FILE → LEAVE**

Look at the files in our new directory. You can use any method you like to browse the directory. To report the directory contents with a DOS command, type the following in the console window:

DIR

Note that the cell file RECROOT.CEL file was created. The journal file for the session was renamed to RECROOT.LOG.

Now reopen the RECROOT cell.

ICRTEST RECROOT

We will now modify the contents of the NAND cell with a nested edit. Use the menu options:

1:**SELECT →** (CELL %)**all → NAND**

2:**EDIT →** (P_EDIT)**select**

You need to reply to a prompt at this point. The editor is warning you that you are about to modify a cell in a protected cell library. You are allowed to edit a copy of this cell because you changed it's protection status to copy-edit on page 301. Respond with a <**L**> to the prompt to continue.

Now add some noticeable text to the cell.

1:(ADD)**text**

Type a short string and press <Enter>. Now return to the RECROOT cell and flag NAND for saving.

1:**FILE → LEAVE**

Now add a box to RECROOT so that the geometry in this cell is also modified, then close the editor and flag RECROOT for saving.

1:(ADD)**box**

1:**FILE → LEAVE**

Now look at the contents of the RECOVERY directory. The modified copy of the NAND cell was saved as NAND.CEL in our tutorial directory. The modified copy of RECROOT was saved as RECROOT.CEL, while the previous version was renamed RECROOT.CL1.

Let us say that the changes to RECROOT went badly, and you are sorry you saved the cell. We will rename the current version of RECROOT to some temporary name (just in case you change your mind) and rename the cell backup to use it as our cell file. Type the following in the console window, or use any method you like to rename the files:

RENAME RECROOT.CEL RECBAD.CEL

RENAME RECROOT.CL1 RECROOT.CEL

Now reopen the editor to edit cell RECROOT.

ICRTEST RECROOT

The box is gone, but the change to the NAND cell is still there. Only the contents of the cell RECROOT were affected by using the cell backup copy. The changes to

NAND were saved in the cell file and this version was the one loaded when the editor was reopened.

The Journal File

The journal file is the key to the recovery mechanism of ICED™. During every ICED™ session, every command executed is also echoed in the journal file. This is true whether commands are typed in or selected from the menus. If for any reason the cell is not saved at the end of an ICED™ session, all your work can be recovered by executing the commands in the journal file. This allows you to recover from system crashes.

The journal file can be modified using any ASCII text editor. The commands in the file can be modified to correct mistakes before using it for recovery. In this manner, the modified journal file can be used to recover from editing mistakes that UNDO cannot reverse (e.g. discovering that you deleted the wrong group of components after executing several more edit commands). We will explore this use of the journal file in a later lesson.

The journal file is created as soon as you launch ICED™. The name of the file is *cell_name*.JOU, where *cell_name* is the name of the cell you launched ICED™ to edit. The journal file will be located in the same directory as the cell file. The journal file is updated as you execute each command. When you terminate ICED™ normally, the file is renamed *cell_name.*LOG. If an old file exists with this name, it is deleted.

Look at the current contents of the RECOVERY directory. Note that the file RECROOT.JOU was just created. The journal file from the previous edit session has been renamed RECROOT.LOG.

If your system crashes, or you terminate ICED™ with the JOURNAL command, the journal file is not renamed. (Under these circumstances, no cell files are saved.) In this case, when you attempt to re-open ICED™ to edit the same root cell, you will be provided the opportunity for automatic recovery.

Automatic Recovery

Let's see the journal file in action. First, add a polygon to cell RECROOT.

1:(ADD)**poly**

If you want, look at the contents of the RECOVERY directory again, and you will see that the timestamp for the RECROOT.JOU file has just changed. The journal file was just updated with the ADD POLY command you just executed.

Now simulate the effects of a power outage by closing the ICED™ window explicitly. Click on the button with the X in the far upper right corner of the window. At the warning prompt, click the JOURNAL button. The JOURNAL command is executed and the editor closes.

Look at the contents of the RECOVERY directory. You can see that the timestamp for cell RECROOT has not changed; the cell file was not saved. You can also see that RECROOT.JOU was not renamed.

If this had been a real power outage, you would have to open a new console window and change to the RECOVERY directory again, but you don't have to do this now. Just make the console window the current window and reopen the editor.

ICRTEST RECROOT

When you launch ICED™, one of the first tasks it performs is looking for a file with the name *cell_name*.JOU. If one exists, you are given the option of performing automatic data recovery. (A file with the name *cell_name*.JO1 will also trigger automatic recovery. .JO1 files are explained below). When a journal file exists, you are presented with a message box similar to the following:

Journal file for Q:\ICWIN\TUTOR\RECOVERY\RECROOT exists.
Do you want to recover?

If you select the NO box at this point (don't do this now), the journal file is renamed to RECROOT.JOX and no work from the last ICED™ session is repeated. (You can still use the .JOX file for recovery later if you change your mind after viewing the cell file. Simply execute the command @RECROOT.JOX from inside ICED™.)

Instead, click the YES box. ICED™ will use the commands in RECROOT.JOU to recover all of your work. The cell will be updated to the state it was before the power was cut off. Note that the polygon is recreated in the cell.

This process is automatic. You do not even have to remember that your work was interrupted. If cells other than RECROOT were modified during the same edit session and not saved by ICED™, they will all be recovered automatically. The only thing you must remember is to edit the same cell as the root cell. **Editing a different cell as the root cell will not trigger automatic recovery.**

Suppose that something goes wrong while ICED™ is performing the recovery. If your system crashes or runs out of hard disk space while ICED™ is processing the journal file, you are still protected.

Before ICED™ starts the recovery process, it renames the *cell_name*.JOU file to *cell_name*.JO1. As ICED™ processes the journal file, it will log all commands as they execute into a new *cell_name*.JOU file. Once the recovery is complete, ICED™ will delete the .JO1 file. However, if the recovery is interrupted, the .JO1 file will still exist. ICED™ detects this when it is launched the next time and will begin the recovery from the beginning again using the .JO1 file.

Recovering From Mistakes

The journal file is simply a list of commands that modified your cell. You can edit these commands before you use the journal file to recover the cell at a state it was in before a mistake was made.

Add a text component to cell RECROOT to simulate work in the current session that you want to keep.

1:(ADD)**text**

Type some string similar to "KEEP THIS" at the prompt, and then add the text component with the mouse.

Now select the polygon, move it, then unselect it with the following menu options:

1:(SELECT)**in**

1:**MOVE → X&Y**

1:(UNSEL)**all**

Now try to undo the move operation with:

1:**UNDO**

Only the UNSELECT operation is reversed by the UNDO. Repeating the UNDO command will only unselect the polygon again. The move operation cannot be undone since a different command was executed after it.

To get the cell back to the way it was before the move operation; you must use the journal file to recover only part of the work done in this session. (We recommend that you backup cell files and the journal file before you begin this type of recovery for real work.) This method consists of the following steps:

1) Use the JOURNAL command to terminate ICED™. No cell files are saved, and the *cell_name*.JOU file is not renamed to *cell_name*.LOG.
2) Edit the *cell_name*.JOU journal file with any ASCII text editor. Starting at the end of the journal file, search upwards for the first command executed in error. Remove the line of the journal file with this command **and all lines following the command down to the end of the file**. (It is important to realize that the state of a cell depends on each command that modifies it. Trying to remove only one command from a journal file and then executing succeeding commands will often lead to unexpected results. The cell is no longer the same cell it was the first time those commands were executed.)
3) Save the modified journal file without changing the file name.
4) Launch ICED™ as usual to edit the **same root cell**. The editor will recognize that data recovery should be performed because the file *cell_name*.JOU exists. The "Do you want to recover?" message will be displayed. Type a <Y> at the prompt and your cell will be restored to the state it was before the bad command was executed.

This process is easier than it appears. Let us perform it now to return the cell to the state it was in before the MOVE command. Close the editor immediately without saving any cell files:

1:**FILE → JOURNAL**

At the prompt, click on the “JOURNAL” button to proceed. If you look at the directory contents, you will see that RECROOT.JOU was not renamed, and the timestamp for the RECROOT.CEL has not changed.

Now edit the file. Type the following in the console window:

NOTEPAD RECROOT.JOU

Fine the line with the MOVE command near the bottom of the file. Delete this line and all following lines to the end of the file. Keep the SELECT command above the MOVE command.

Save the file and exit Notepad.

Now reopen the layout editor with exactly the same command you used to open it previously:

ICRTEST RECROOT

At the dialog box that asks you if you want to recover, click on “YES”. Once the editor is open, you will see that your text component has been recreated, but the polygon is back at its original position. Your cell is back at the state it was in before the MOVE command. You could continue to work in this cell as usual.

Journaling During Command Files

When executing a command file, you have the choice of logging just the command that calls the command file, or logging each command in the file. It is recommended to log each command in the command file. (This is due to the fact that recovery of the data is guaranteed if all commands are logged into the journal file. If only the command that called the command file is logged in to the journal file, and you edit the command file before recovering, the edited command file will be executed instead of the original command file.) Logging each command in the command file is the default unless a LOG OFF command is used as the first command in a command file.

If the LOG mode is not turned off, the commands in the command file are stored in a buffer and dumped into the journal file as the buffer fills. (This is much faster than opening and closing the file for each command.) When the command file is completed, or when an error is encountered, any commands remaining in the buffer are dumped into the file.

If your system crashes during the command file, some commands may not be written to the file. If you use the journal file for recovery, you should always remove all commands from the end of the file up to the command that called the command file. It is important to execute an entire command file as a unit.

Recovery After Cell Files Are Saved

Even once the editor is closed and cell files are saved, it is still possible to recover from errors made during the last ICED™ session. The basic procedure is as follows:

1) Rename or move all cell files saved in the last session from the current directory. (You can delete them later once you are happy with the recovery effort.)
2) Rename the cell backup files to the original cell names. At this point your cell files are restored to what they were before the last edit session
3) Edit the *root_cell_name*.LOG journal file from the last session with any ASCII text editor. Starting at the end of the journal file, search upwards for the first command executed in error. Remove the line of the journal file with this command **and all lines following the command down to the end of the file**.
4) Save the modified journal file to a new command file name.
5) Launch ICED™ as usual to edit the same root cell. No automatic recovery will be performed since *root_cell_name*.JOU does not exist. Now execute the modified journal file as a command file and all cells will be restored to the state they were in before the mistakes happened.

First, let us perform some work we want to preserve in each cell. We will add the text "Good work" to each of the open cells. To do this, we will use the _LOOP2.CMD command file introduced in the Creating Useful Command Files tutorial on page 254. This command file executes a command string in all open cells, including all subcells. Type the following as a single command on the command line. (This means that the "…" indicators are not typed. This is just syntax that allows us to fit long command lines on the printed page.) Be sure to type the different quote characters as shown.

@_LOOP2.CMD;#LOOP.OP="ADD TEXT='GOOD WORK' …
… AT 10,10";#LOOP.LEVEL=2

(Note that the extra text in the NAND cell is due to the fact that copies of cells VIA, NN, and PP are used inside the NAND cell. These cells had text components added to them as well.)

Now let us perform some work in each cell that we will consider as mistakes and remove later. Use the command history feature (the <↑> key) to retrieve the last command, then edit the command line before pressing <Enter>.

@_LOOP2.CMD;#LOOP.OP="ADD TEXT='**UH OH**' **…**
… AT **20,20**";#LOOP.LEVEL=2

Now save these modifications and close the editor:

1:**FILE → LEAVE**

Now open the editor again.

ICRTEST RECROOT

Be sure to yell "UH OH!" when you see those changes made in error. You must be careful to not save any files at this point, or the backups we need will be overwritten. Close the layout editor without saving any files:

1:**FILE → JOURNAL**

Now look at the contents of the RECOVERY directory. Switch to the console window and type:

DIR

A RECROOT.JOU file was created when we opened the layout editor last. Delete this file to prevent automatic recovery of the session where all we did was yell "UH OH!".

DELETE RECROOT.JOU

The journal file from the edit session where you made the errors is stored as RECROOT.LOG. Open this file with Notepad.

NOTEPAD RECROOT.LOG

Scroll to the end of this file. You should see the list of cells saved during that session. The list will look something like the following:

```
! Saving Q:\ICWIN\TUTOR\RECOVERY\RECROOT.CEL
! Saving Q:\ICWIN\TUTOR\RECOVERY\NAND.CEL
! Saving Q:\ICWIN\TUTOR\RECOVERY\NN.CEL
! Saving Q:\ICWIN\TUTOR\RECOVERY\PP.CEL
! Saving Q:\ICWIN\TUTOR\RECOVERY\VIA.CEL
```

First, you need to move all of the cell files saved with errors to a new subdirectory. If this was a real design and you had modified only some of the cell files in the current directory, **you should be careful to move (or rename) only the cell files saved by this bad session**. However, in our case, the only cell files in the directory are those saved by the last session, so we can just move all cell files to a temporary place where they can be easily deleted after a successful recovery.

MD BAD

MOVE *.CEL BAD

DIR

The only files left in the directory should be cell backup files for NAND and RECROOT. No cell backups exist for cells NN, PP, and VIA. This is because the original versions exist in the SAMPLES cell library. When the editor opens, it will find the original cells there. However, the RECROOT and NAND cell backups need to be renamed to their original cell names.

RENAME RECROOT.CL1 RECROOT.CEL

RENAME NAND.CL1 NAND.CEL

To recover part of the work done during the session where you made the errors, you edit the journal file. Remove the commands executed in error and all commands after them. Starting at the end of the file and looking up, find the line in the Notepad window where you executed _LOOP2.CMD the second time. It should be indicated with comment lines that look like the following:

```
! @Q:\ICWIN\AUXIL\_LOOP2.CMD
! LOCAL #LOOP.OP="ADD TEXT='UH OH' AT -20,-20"
```

Starting with these two lines, delete everything from this point to the end of the file. Now save the file to a new name, **RECOV.CMD**. (Use File → Save As.) Then close the Notepad window.

Open the layout editor again to edit the same root cell.

ICRTEST RECROOT

Now re-execute the commands executed in the previous session before the errors occurred by executing the command file RECOV.CMD.

@RECOV

The “GOOD WORK” text components have been re-created but the “UH OH”s have not been re-created.

You should take care with this type of procedure for real design work. It is best to perform this type of recovery in a copy of your design directory, rather than in the original directory. Then overwrite the cell files once you are happy with the results. You should at least back up all cell libraries before attempting a recovery like this.

Conclusion

The journal file is a powerful mechanism for data recovery. It is flexible, so it places control of a recovery operation in your hands, rather than limiting your options. Now that you have used it successfully in several situations, you have a powerful tool to recover your work from many types of editing disasters.

Index