Drawing in Magic

Magic is built around one idea, and once you have it the rest is mostly vocabulary: **there is a box, you move the box, and you tell a layer to fill the box.** Magic calls that painting, and everything else you do, including contacts, labels, ports and zooming, is done either to the box or with it.

This page explains the commands you'll use from Lab 02 onwards, and you'll draw your first cell with them in Lab 03.

The mouse

Left	move the **lower-left** corner of the box to the cursor
Right	move the **upper-right** corner of the box to the cursor
Middle	**paint** the box with the layer under the cursor, or **erase** if the cursor is over empty space

So to draw a rectangle you left-click one corner, right-click the opposite corner, and then middle-click a swatch of the layer you want in the palette down the right side of the window. There is no click-and-drag. The first hour of working this way is annoying, but after that it's faster than dragging would be.

If you have no middle button, or you want exact coordinates, you can type the command instead. In this course you always want exact coordinates, so typing is how you'll draw most of the time.

The console

Magic's second window, `tkcon`, is the console where you type commands, and every command on the rest of this page is typed there.

```
box 0.4um 1.3um 2.6um 1.95um     ;# put the box at these micron coordinates
paint ndiff                      ;# fill it with n-diffusion
erase ndiff                      ;# or take it away again
```

Use the `um` suffix. Without it, Magic reads a number in *lambda*, which is 0.01 µm in this technology, so a box you meant in microns comes out a hundred times too small, and Magic gives no error message about it. Every command in every lab writes `um`, and yours should too.

Two more commands you'll use constantly:

```
box values                       ;# what is the box now? (in internal units, 0.005 um)
findbox zoom                     ;# zoom the view to the box
```

`box values` **reads the box back in internal units of 0.005 µm**, even though a plain number you give to `box` *sets* it in lambda. That inconsistency has cost people whole afternoons. The safe habit is to always set the box with `um`, and to read the numbers `box values` returns as 200 to the micron.

Layer names

The palette on the right shows each layer's display name, but the console takes short aliases instead. These are the ones this course uses:

Console name	Palette	What it is
`nwell`, `pwell`	nwell, pwell	the wells

Console name	Palette	What it is
`ndiff`, `pdiff`	ndiffusion, pdiffusion	source/drain silicon
`ntap`, `ptap`	nsubstratendiff, psubstratepdiff	well and substrate taps
`ndc`, `pdc`	ndcontact, pdcontact	contact from diffusion up to li
`ntapc`, `ptapc`	nsubstratencontact, psubstratep-contact	contact from a tap up to li
`poly`	polysilicon	the gate layer
`pc`	pcontact	contact from poly up to li
`li`	locali	local interconnect
`mcon`	viali	contact from li up to metal1
`m1`, `m2`	metal1, metal2	wiring
`via1`	m2contact	metal1 to metal2
`pfethvt`	pfethvt	makes a PMOS the high-threshold flavour

`tech layers` prints the full list with every alias.

Contact layers

Contacts surprise everyone who is new to Magic, because you don't place a contact as an object. Instead you **paint the contact layer over an area**, and Magic treats that region as the whole stack at once: the diffusion, the cut, and the local interconnect on top. So `paint ndc` over a 0.17 × 0.17 µm box gives you one properly formed n-diffusion contact.

That has a few consequences, and it's worth knowing them before they confuse you:

- For a single cut, the contact region must be **exactly 0.17 µm** in each direction (rule `licon.1`). If you make it bigger, Magic generates an array of cuts instead.
- The plain layers still have to surround the contact. The diffusion has to extend past the contact, and so does the li. The rules for those are `licon.5c` and `li.5`, and `li.5` is one of the five rules you fix in Lab 02.
- Erasing `li` from a contact doesn't remove the contact; Magic demotes it instead.

Labels and ports

As far as LVS is concerned, a cell has no terminals until you name them, and there are two commands that give things names:

```
box 1.30um 2.40um 1.55um 2.65um
label A c li          ;# put the label "A", centred, on the li layer here

select top cell
port makeall          ;# promote every label in the cell to a port
```

`label` names a piece of geometry, and `port makeall` then declares those names to be the cell's terminals. That second step is what makes the extracted netlist come out as `.subckt my_inv A Y VPWR VGND` rather than as a subcircuit with no pins, so every lab step that adds labels ends with `port makeall`. If netgen ever says a cell's pin list is empty, a missing `port makeall` is the first thing to check.

A label always goes into the cell you're editing, on that cell's own geometry under the box. If the only shape under the box belongs to a subcell, the label still lands in your cell, but on empty space: Magic prints `Moving label "A" from locali to space`, and a label on space isn't attached to any net, so it never becomes a pin on the extracted `.subckt` line. That's why the blocks in Lab 04 and the capstone paint their own small patch of `li` or `m1` before labelling it.

The `.mag` file

```
save my_inv           ;# writes my_inv.mag in the current directory
```

A `.mag` file is plain text, made up of one section per layer, each listing that layer's rectangles:

```
<< ndiff >>
rect 80 260 520 390
```

Check the header before you trust those numbers. The unit a `.mag` file uses is set by its `magscale` line, and the two kinds of cell in this course use different units:

Cell	Header	One unit is	`rect 80 260 520 390` means
A foundry cell from the PDK	`magscale 1 2`	0.005 µm	0.40 ⟶ 2.60 µm
Anything this course draws	*no `magscale` line*	0.01 µm	0.80 ⟶ 5.20 µm

So the rectangle above spans 0.40 ⟶ 2.60 µm horizontally if it's **in a foundry cell**. In your own `mag/my_inv.mag`, the same 0.40 ⟶ 2.60 µm diffusion is stored as tiles spanning `40 130` to `260 195`. Magic splits a painted shape into several `rect` lines, so read the extent of the whole shape rather than any one line. If you apply the foundry cell's unit to a cell you drew, you'll read every dimension as half its real size, and nothing will warn you; it's the same kind of factor-of-two trap as `box values`. Magic does tell you which kind of cell you have every time it loads one (`Scaled magic input cell my_inv geometry by factor of 2`), and `box values` always reports in 0.005 µm units, whichever kind of file you're in.

When a lab says that a shipped cell has a particular dimension, the number comes from the cell's `.mag` file, so you can check it yourself with `grep`. In Lab 03, Part 9 you open your own `.mag` file in a text editor.

The commands the labs run

Later guide pages cover each of these commands, and together they make up the whole flow:

```
drc check                 ;# find rule violations
drc why                   ;# name the rule broken under the box
drc style drc(full)       ;# switch to the rule set that catches taps

extract do local          ;# geometry -> circuit
extract all
ext2spice lvs             ;# write it in a form netgen understands
ext2spice -o my_inv.spice
```

```
ext2spice cthresh 0       ;# ...and keep every parasitic capacitance
gds write my_inv.gds      ;# the file a foundry would take
```

The Magic cheat sheet lists these with their keyboard equivalents, plus about a dozen more.