**L-Edit tutorial**

**1. Getting started**

The Layout contains the hierarchy will all the cells you create. When saving you save the entire Layout and not just a specific cell. Think of the Layout as the project.

For a new Layout, click File->New. For "File type" choose "Layout" and from the list below choose "<empty>".

For an existing Layout, choose View->Design Navigator to see the T-cells associated with the layout and choose a file to edit, or click on the Cell to see the design.

On the left side of the screen there's a box with the layers. Right-click on it and choose "Setup layers". Here you can add layers and give them names and color codes (under the "Rendering" tab). Layers that will be written (and are not auxiliary layers for programming purposes) need a CIF name. This name will be associated with the layer at the lithography stage, so make it meaningful.

Choose Cell->New to design a new cell. Give it a meaningful name. Go to the T-Cell Parameters tab and define parameters. You can skip this and code everything directly, but if you want a design parameter to be determined by a different cell sitting higher in the hierarchy it must be defined here. Good practice: define a String variable "Layer\_name".

After clicking 'Ok' the program displays the code of the cell.

**2. Writing design code**

This sections is divided into three:

Lines you should copy/paste,

How to write design code,

How to use layers,

How to write code for higher levels in the hierarchy

**2.1. Lines you should copy/paste**

Locate the sections:

**/\* Begin -- Remove this block if you are not using L-Comp. \*/**

**#include "lcomp.h"**

**/\* End \*/**

And below it paste the following:

**const int PARAMSMAX = 10000;**

**void booleanOp(LCell pCell, LBoolean boolOp, LLayer layer1,**

**LLayer layer2, LLayer resLayer, LBoolean delInputs) {**

**int i = 0,j = 0;**

**LObject l1[PARAMSMAX], l2[PARAMSMAX];**

**LObject obj;**

**int n1 = 0, n2 = 0, no = 0;**

**obj = LObject\_GetList(pCell, layer1); // Get all objects on layer1**

**while (obj != NULL) {**

**l1[i++] = obj;**

**obj = LObject\_GetNext(obj);**

**}**

**obj = LObject\_GetList(pCell, layer2); // Get all objects on layer2**

**while (obj != NULL) {**

**l2[j++] = obj;**

**obj = LObject\_GetNext(obj);**

**}**

**// perfrom operation**

**LCell\_BooleanOperation(pCell, boolOp, 0, l1, i, l2, j, resLayer, delInputs);**

**}**

This is necessary to merge different layers.

Next locate the following section:

**/\* Begin -- Remove this block if you are not using L-Comp. \*/**

**LC\_InitializeState();**

**LC\_CurrentCell = cellCurrent;**

**/\* End \*/**

And below it paste the following:

**LFile pFile = LCell\_GetFile(LC\_CurrentCell);**

**LLayer layer = LLayer\_Find(pFile, Layer\_name);**

Include the second line only if you defined such a T-cell parameter as recommended in Sec. 1.

Any parameters which are not being passed to other cells but are instead intended to be written directly must be converted to "internal units". This part of the software is impossible to understand so just stick to the manual.

Suppose you defined a T-cell parameter "R\_s". After the previous block of code, paste the following:

**R\_s = LFile\_DispUtoIntU(pFile, R\_s);**

Do this for every T-cell parameter which is not going to be passed to other T-cells.

**2.2 Writing design code**

Break up your design into boxes, other polygons, tori, pies and circles. Here I show explicitly how to use the first three.

These objects accept different parameters. All of them need an LPoint object, so define an array of class LPoint:

LPoint pts[10];

Each element has an x and y coordinate. The following statements are legal:

**pts[5].x=10; pts[5].y=20;**

**pts[5].x=10; pts[5].y=pts[4].y+10;**

**pts[5]=pts[2];**

To draw a box on layer "Layer\_name" with opposing corners at pts[0] and pts[1], write:

**LC\_CreateBox(Layer\_name, pts[0], pts[1]);**

Note that Layer\_name is a String object, NOT a LLayer object. You can give the box a name like this:

**myBox = LC\_CreateBox(Layer\_name, pts[0], pts[1]);**

You can also define an array

**LObject myBoxes[10];**

**myBoxes[5]= LC\_CreateBox(Layer\_name, pts[0], pts[1]);**

This looks neater, but I haven't found this to be very useful. Theoretically this would allow another T-cell to obtain the parameters of myBoxes[5] and do something with them.

To write a torus, first you must define a LTorusParams object:

**LTorusParams tParams;**

**tParamsI[1].nInnerRadius = R–w/2;**

**tParamsI[1].nOuterRadius = R+w/2;**

**tParamsI[1].dStartAngle = 270;**

**tParamsI[1].dStopAngle = 0;**

**tParamsI[1].ptCenter = LPoint\_Set(pts[1].x,pts[1].y);**

The angle starts on the positive x axis and increases counter-clockwise.

Write the torus:

**LTorus\_CreateNew(LC\_CurrentCell, layer, &tParams);**

Note how this differs from LC\_CreateBox. Here "layer" is an LLayer object, NOT a String object.

To write a general polygon, define an LPoint array with as many elements as you want vertices. Create the polygon with:

**LPolygon\_New(LC\_CurrentCell, LayerName, pts, 8);**

Here "LayerName" is not a previously defined object; if the layout has a layer named "LayerName" the polygon will be drawn there. The number 8 means only the first 8 elements of the LPoint array "pts" define the polygon and the rest of the array is ignored.

**2.3 Working with layers**

Even a simple design can require several layers which are merged into one or two final layers. Be sure to give the final layers meaningful names so they can be identified in the fabrication process; the others can have names like temp2 or t3.

To merge layers, copy the following into the end of the T-cell code (before the two { symbols at the bottom):

**LCell\_SetLock(LC\_CurrentCell, 0); // Can only operate on flattened cells.**

**LCell\_Flatten(LC\_CurrentCell);**

**booleanOp(LC\_CurrentCell, LBoolOp\_XOR,**

 **LLayer\_Find(pFile, "t1"),**

 **LLayer\_Find(pFile, "t2"),**

 **LLayer\_Find(pFile, "t4"), LFALSE);**

**booleanOp(LC\_CurrentCell, LBoolOp\_XOR,**

 **LLayer\_Find(pFile, "t3"),**

 **LLayer\_Find(pFile, "t4"),**

 **LLayer\_Find(pFile, "Base"), LFALSE);**

This code take layers t1 and t2 and XORs them together to make layer t4. Then layers t3 and t4 are XORed together to produce layer Base. All these layers must be named in the layout beforehand; the code doesn't define new layers but works with existing layers.

Concrete example: a coplanar waveguide. One layer is the ground plane, which is just a box in layer t1. To make a coplanar waveguide with a gap to the ground plane, define the waveguide on layer t3 and a wider waveguide (with width waveguide+2\*gap) on layer t2. XORing layers t1 and t2 to produce layer t4 will result in a ground plane with gaps of the necessary size, and then XORing layers t3 and t4 will place the waveguides within the gaps to produce layer Base.

**2.4 Building a hierarchy**

Suppose you have a T-cell named "Junction" which you want to place over and over again in the design, possibly changing a parameter each time. You will need to pass a pointer to the T-cell parameter values you wish to use, so first write:

**char \*params[PARAMSMAX];**

**int i=0;**

**params[i++] = "s"; params[i++] = LFormat("%f", s);**

**params[i++] = "h"; params[i++] = LFormat("%f", h);**

**params[i++] = "b"; params[i++] = LFormat("%f", b);**

**params[i++] = "N"; params[i++] = LFormat("%d", N);**

**params[i++] = "Layer\_name"; params[i++] = "JunctionLayer";**

**params[i++] = NULL;**

Here the T-cell parameters s, h, and b expect a float, N expects an integer and Layer\_name expects a string. Now write:

**LC\_SetXYPlacementPosition(0, 0);**

**LC\_Generate("Junction","Junction001",params);**

This instances the T-cell "Junction" with parameters "params" at coordinates (0,0) and gives it the name "Junction001" to let it be addressed by other cells (the instance name does not usually come in handy so don't worry about it).

When using LC\_SetXYPlacementPosition remember that it uses internal units. If using units which have not been converted, do this:

**LC\_SetXYPlacementPosition(LFile\_DispUtoIntU(pFile, Corner\_1), LFile\_DispUtoIntU(pFile, Corner\_2));**

The following can be used to set the new position relative to the current position:

**LC\_IncrementXPlacementPosition(LFile\_DispUtoIntU(pFile,R\_spacing));**

Here's an example:

**char \*params[PARAMSMAX];**

**int i=0;**

**int j;**

**LC\_SetXYPlacementPosition(0, 0);**

**for (j=0; j<10; j++)**

**{**

**params[i++] = "s"; params[i++] = LFormat("%f", s);**

**params[i++] = "h"; params[i++] = LFormat("%f", h);**

**params[i++] = "b"; params[i++] = LFormat("%f", b);**

**params[i++] = "Layer\_name"; params[i++] = "JunctionLayer";**

**params[i++] = "N"; params[i++] = LFormat("%d", N);**

**LC\_Generate("Junction","junction",params);**

**LC\_IncrementXPlacementPosition(LFile\_DispUtoIntU(pFile,3\*s));**

**h=h+h/10;**

**}**

To instance a rotated T-cell, write:

**LC\_SetPlacementOrientation(LRotate180);**

**LC\_GenerateAlign("Junction", "junction", params);**

Note LC\_GenerateAlign and not LC\_Generate. For more orientation options consult the manual.

Merge layers as far up in the hierarchy as you can, so each merging action is performed only once. Place as many instances as needed on as many layers as needed, and merge the layers only once at the end, and not for each instance.

Display units vs. internal units: a parameter passed to the cell is in display units and must be converted to internal units. If the T-cell has a parameter "w" which you choose to equal 10 when the cell is instanced, you need to convert it like this:

**w = LFile\_DispUtoIntU(pFile, w);**

(after defining pFile as given earlier)

Same for values that are passed in the params list above. The only values which are in internal units are values defined within the cell. For example, "int w=5" gives a variable w whose size is in internal units.

Bottom line: only the T-cells which write directly on the main cell should work with internal units, i.e. convert every parameter they accept with LFile\_DispUtoIntU, or write directly with locally defined variables. Every higher level on the hierarchy which passes parameters to lower levels should pass them with display units. (Try not to define local variables within higher levels of the hierarchy to pass to lower levels; these will be in internal units.)

**3. Exporting the design**

To be completed.