EAGLE CAD layers reference

rev. 1.0 – March 2011

Introduction

This document can be used as a layers reference for those looking to create PCBs with EAGLE CAD. The tutorials and help content that is supplied with EAGLE are both excellent, but they are missing some explanations on the layers so this guide can supplement it.

The layers are explained through the use of examining through-hole components, surface mount components, the PCB legend(s) and the edges of the board. Finally, there are working examples of component construction with a focus on the layers. The working examples are supplied with source code for easy replication.

Contents

Components overview

At a birds-eye view, components typically contain some type of pads, and a legend for the PCB. The layers are used to define all this in detail, so that the steps to manufacture the PCB are catered for, with all the information that is needed.

PCB legend

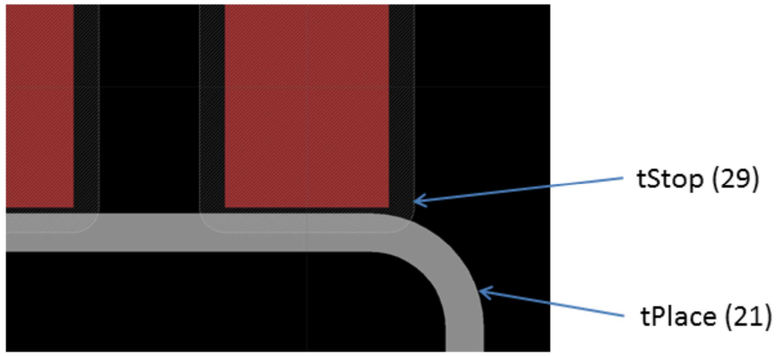
The PCB legend is white text can be printed onto the PCB. This is useful for showing where the components will be placed, and also the identifier (resistor R1, R2 etc) or component value (the default CAM processor settings don't print the component value). Some manufacturers can print legends on both sides of the board.

The default setting is to print a top legend of tNames, tPlace and Dimension layers. (The Dimension layer is just the outline of the board). Some symbols show the tPlace legend trampling over the SMD pad, but the manufacturer will automatically override this because of the tStop layer.

The tStop layer is an outline which is used to determine where the green insulating colour on the PCB will not occur (i.e. the pad will be copper/silver, with no green insulator).

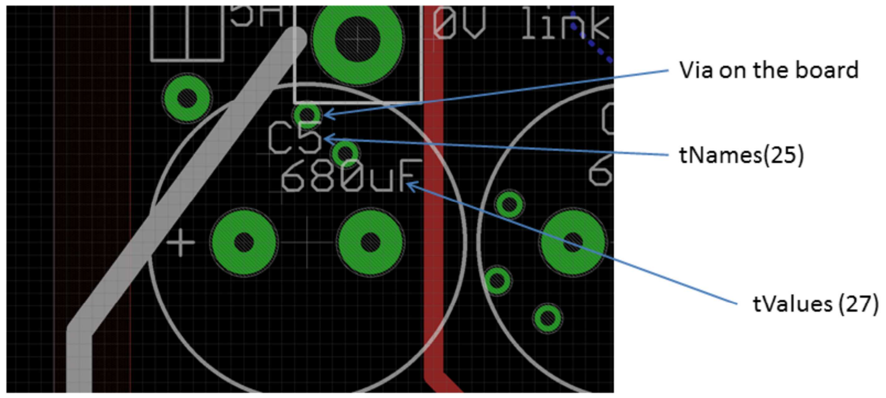
The tStop layer always takes precedence.

Here is an example. Here, the tPlace layer clearly transgresses the tStop layer. This is ok, the manufacturer will just not print the legend in the whole tStop area. It protects the pad. In general, it can therefore be assumed that no legend can ever accidentally cover a pad.

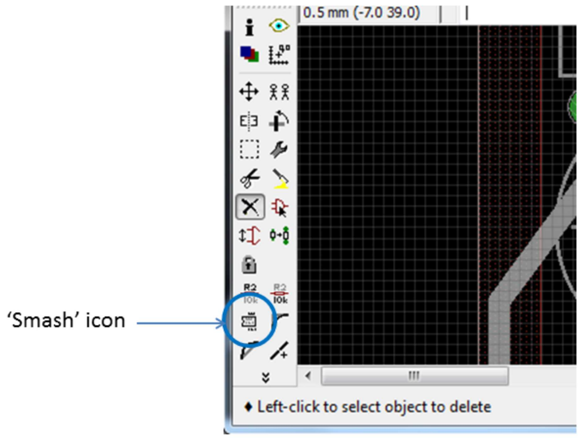


To summarise, it can be seen that the manufacturer will print a legend taking into consideration the Stop layer.

The next example shows the component identifier (tNames layer) and component value (tValues layer) for a capacitor. It can be seen that there are some vias on the board, and the text is trampling over the vias. There is no problem electrically, because the tStop area around the via will protect it from the legend. However visually, the board text legend will have a butchered '5' in the text 'C5'. To fix this for visual reasons, the C5 text should be moved. Note there is no need to move the '680uF' text (even though that is trampling over a via too), because the tValues layer is not usually part of the legend anyway. If you wish to make it part of the legend, then ideally it should be moved, to prevent a butchered digit '0' in '680uF' in this example.

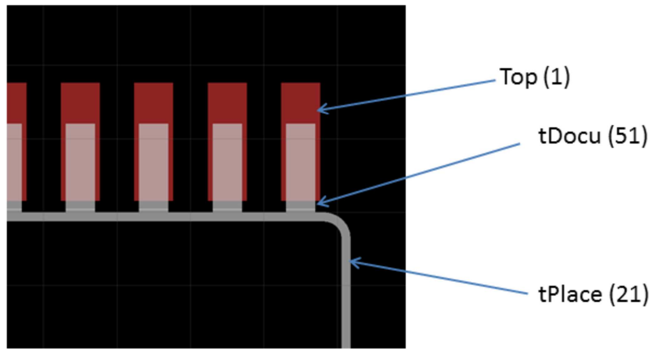


Component text is moved by using the 'smash' icon and then selecting the component. It will separate the combined component.



Left-click to select object to delete

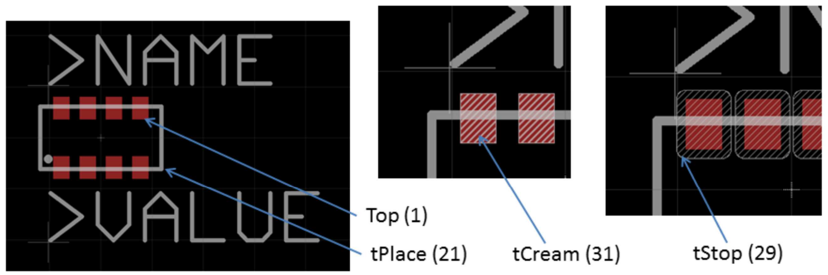
Here is something else that is common in many of the packages. The curved line in the image below is the usual tPlace layer, but the rectangle covering the pad is the tDocu layer. The tDocu layer is generally a useless layer. It is not part of the PCB or the legend, so the fact that it is trampling the pad can always be safely ignored. The tDocu layer is used only when documentation is printed, to show where the component goes.



Surface mount components

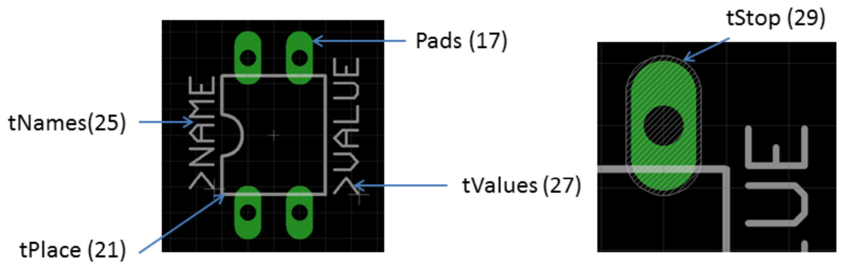
The cream layer is the solder paste layer, i.e. the area where the paste would be applied (before the components are placed on the board) in a manufacturing process.

The stop layer is the same as described earlier, i.e. it is the area which will not be coated green.



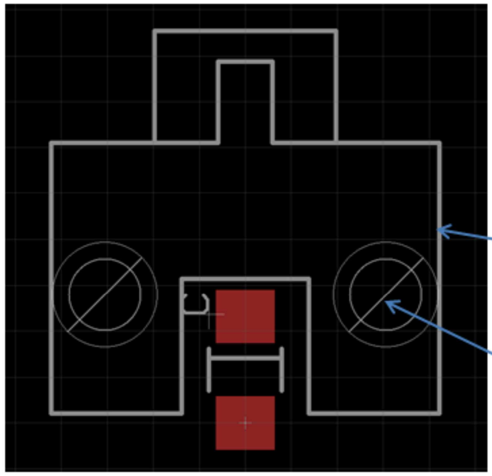
Through-hole components

The pads automatically contain drill holes and the 'tStop' outline which is used to determine where the green insulating colour on the PCB will not occur.



Holes

Components may need holes; here is an example of an LED with a plastic light pipe fitting. Holes can also be placed directly on the circuit board. When a hole is placed, it can be seen by viewing the Holes layer. Note that there are different symbols for different sized holes, so visually on the screen you may see a different shape from the one in this example. Regardless of the symbol on the screen, the final drilled hole is the same of course, it is just a hole.



Note that there is a circle for the hole symbol in the Dimension layer too, and this can be used to see the actual circumference of the drilled hole.

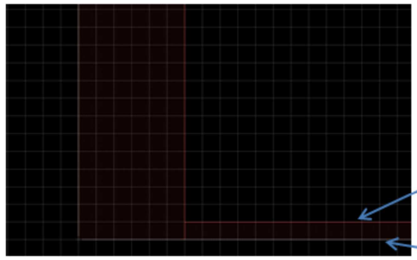
PCB outline and protected areas

The overall PCB shape is defined using the Dimension layer. Usually it is good to ensure that no components are placed too close to the edge of the board, and that no tracks are drawn too close to the board. These two functions are achieved using the tKeepout (banned area for component placement) and tRestrict (banned area for tracks) layers respectively. The tKeepout and tRestrict layers can also be used elsewhere too, their usage is not limited to the edge of the PCB. There is also a vRestrict layer for vias (banned area for vias). In summary it is generally wise to place these layers at least on the edges of the board.

The diagram here shows the bottom-left corner of a PCB. The outline of the PCB is drawn in the Dimension layer with a width of zero.

For the Restrict/Keepout layers, five rectangles (all on top of each other effectively) were drawn for each side of the PCB. These five layers prevent components from being placed too close, it prevents vias and also tracks.

Note that the rectangles on the left edge of the board are quite wide. This is because this board was intended to be slid into a grooved metal enclosure along that edge. The metal groove was required to be kept totally away from components, vias and tracks.



Component creation process

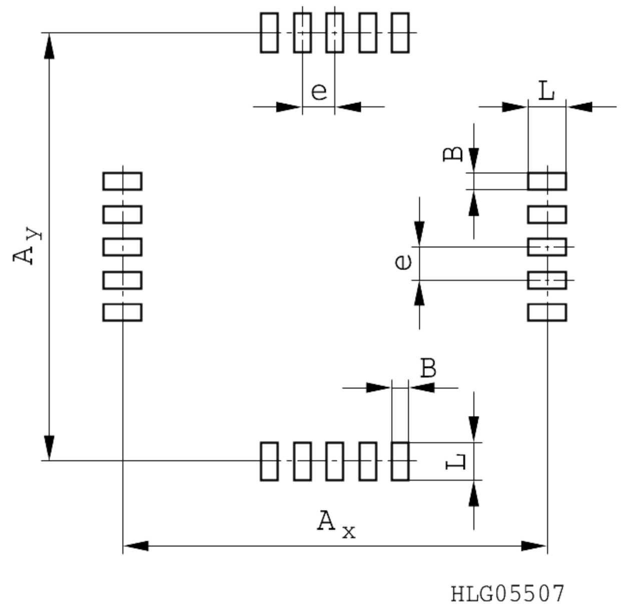
The component creation process is well described in other documents, so this document will only touch on some aspects concerning layers, and will not focus on the schematic symbol nor the pad and electrical pin naming, etc.

Most designs will need some component created, and this document focuses on one surface mount component and one through-hole component.

1. Lay down the pads

The pad size and location are of course highly important. Sometimes bits of an existing component can be reused. If this is not possible, it is best to consult the component data sheet.

Here is an example of the TQFP-48 package recommended landing pads, from Infineon's website:



Dimensions:

Ax = 8.45	**Ay = 8.45**	**e = 0.50**	**L = 1.35**	**B = 0.29**

From this diagram and dimensions there is sufficient information to construct the pads in EAGLE.

It is also possible to automate the component construction. An example for constructing the TQFP package pads is shown in the appendix.

The code is entered into a computer, and then the first few lines can be edited based on the data sheet:

```
#define PWIDTH 0.29    /* this is the pad width in mm */
#define PHEIGHT 1.35   /* this is the pad height in mm */

#define AX 8.45
#define e_val 0.5
#define pins 48
```

The code lines above clearly show how the information from the data sheet can be mapped into a computer program.

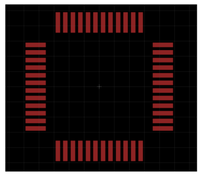
On a Linux server, the program can be compiled and then run.

```
g++ tqfp48.c -o tqfp48_executable
./tqfp48_executable > tqfp48.scr
```

This will generate a tqfp48.scr file which is an EAGLE script file. The first few lines of the generated files contain this:

```
grid mm
grid 1
smd 0.290000 1.350000 (-2.750000 -4.225000)
smd 0.290000 1.350000 (-2.250000 -4.225000)
smd 0.290000 1.350000 (-1.750000 -4.225000)
smd 0.290000 1.350000 (-1.250000 -4.225000)
... etc
```

When this script is run in EAGLE (Windows or any other version) it will construct the pads as shown here.

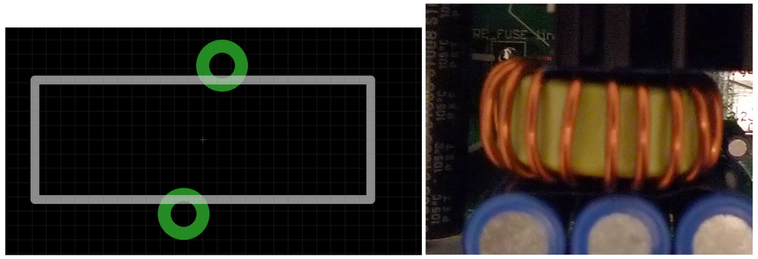


The code could be extended to also draw an outline on the tPlaces layer.

2. Draw the outline

The component outline is drawn in one of the layers that will make it into the legend, i.e. the tPlaces layer.

Here is an example for a through-hole component; a toroid inductor mounted vertically.



Note that even though the tPlaces layer tramples over the pads, this does not matter since the pads always have a tStop outline to protect them.

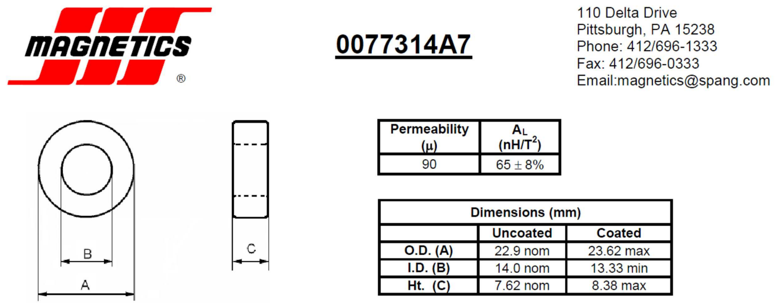
A program to automate this is shown in the appendix. It can draw toroids for mounting horizontally or vertically.

Here is the specification for a toroid.

By modifying the first few lines, it is possible to select the toroid dimensions, and the number of turns of wire on the toroid.

```
// *************** CONFIG *********************************

// style 0=vertical mount, 1=horizontal (flat mount)
#define STYLE    1

// drawing thickness (0.6mm should be always fine)
#define LTHICK  0.6
// toroid dimensions
#define T_ID  13.3     // inside diameter in mm
#define T_OD  23.6     // outside diameter in mm
#define T_WIDTH 8.38   // toroid thickness in mm

// wire details
#define W_THICK   1.626       // 1.626mm is 16 SWG
#define TURNS     18          // inductor will have 18 turns

// wire minimum bend radius
#define MINB    1       // 16 SWG wire will need 1mm to bend
// spacing of turns from each other (in mm)
#define GAP    0       // trial and error, start with zero
// *******************************************************
```

The settings above are for an inductor based on the data sheet of this toroid core:



MAGNETICS®
B
A
C

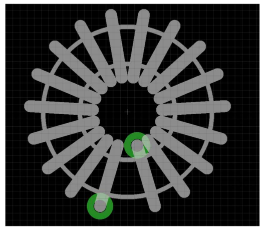
0077314A7

110 Delta Drive
Pittsburgh, PA 15238
Phone: 412/696-1333
Fax: 412/696-0333
Email:magnetics@spang.com

Permeability (μ)	A_L (nH/T^2)
90	65 ± 8%

Dimensions (mm)		
	Uncoated	Coated
O.D. (A)	22.9 nom	23.62 max
I.D. (B)	14.0 nom	13.33 min
Ht. (C)	7.62 nom	8.38 max

An example horizontal-mount toroid generated with these settings would look like the following:



3. Label the part

The last essential step is to assign placeholders for the component identifier and component value, in the tNames and tValues layer respectively. This is best done manually and not automated, since there may be a preference to locate it in a nice position visually. The placeholder text must be **>NAME** and **>VALUE** respectively.

The placeholder text for the component identifier should ideally not trample over pads, since visually it will be unreadable when the board is manufactured.

4. Optional steps

If no tracks or vias are desired under certain areas of the component, then the tRestrict and vRestrict layers should be used to define the region. This is an important step for certain components due to interactions caused by inductance for example.

It may be desirable to place an outline in the tDocu layer. It can be similar to the one in the tPlaces layer if desired, or a simplified version. This is not an essential step by any means.

Appendix

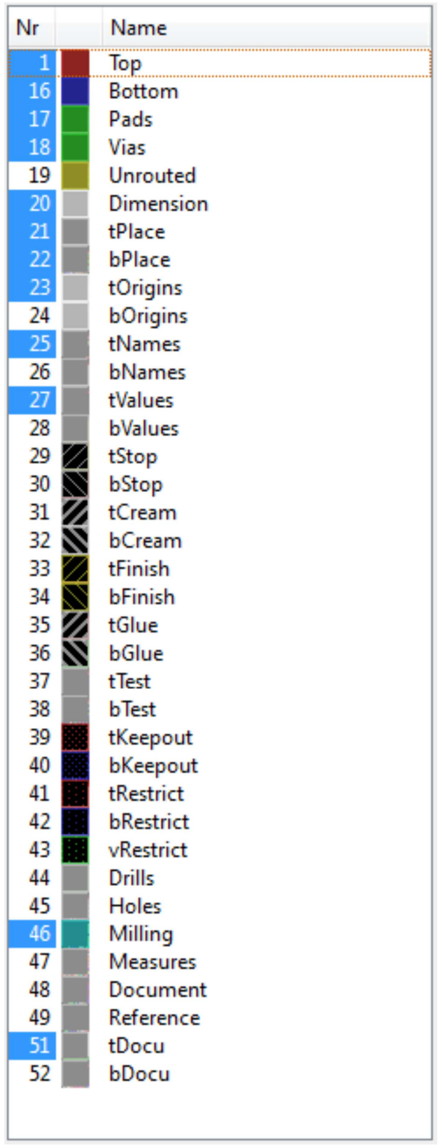
TQFP-48 example code

```
// TQFP drawing tool v1.0
// Revision log:
// Version 1.0
//
#include <stdio.h>

#define PWIDTH 0.29
#define PHEIGHT 1.35

#define AX 8.45
#define e_val 0.5
#define pins 48



int
main(void)
{
  int i, j;
  double x, y;
  double p, np;
  int ti;
  double ax2, nax2;
  int sc;
  int iit;

  printf("grid mm\n");
  printf("grid 1\n");

  p=pins/4;
  p=p/2;
  ti=(int)p;
  p=(double) ti;
  p=((p-1)*e_val)+(e_val/2);

  iit=pins/4;
  if (iit % 2 == 0)
  {
        // even number, nothing else to do
  }
  else
  {
        p=p+e_val/2;
  }

  np=0-p;
  ax2=AX/2;
  nax2=0-ax2;

  sc=pins/4;


  x=np;
  y=nax2;
  for (i=0; i<sc; i++)
  {
    printf("smd %f %f (%f %f)\n", PWIDTH, PHEIGHT, x, y);
    x=x+e_val;
  }

  x=ax2;
  y=np;
  for (i=0; i<sc; i++)
  {
    printf("smd %f %f (%f %f)\n", PHEIGHT, PWIDTH,  x, y);
    y=y+e_val;
  }

  x=p;
  y=ax2;
  for (i=0; i<sc; i++)
  {
    printf("smd %f %f (%f %f)\n", PWIDTH, PHEIGHT, x, y);
    x=x-e_val;
  }

  x=nax2;
  y=p;
  for (i=0; i<sc; i++)
  {
    printf("smd %f %f (%f %f)\n", PHEIGHT, PWIDTH, x, y);
    y=y-e_val;
  }



  return(0);

}
```

Toroid inductor example

```
// toroid drawing tool v1.0
// Revision log:
// Version 1.0
//

#include <stdio.h>
#include <math.h>

#define PI 3.141592654

#define DEG90   1.5707963
#define DEG180  3.141592654
#define DEG270  4.71238898
#define DEG360  6.28318531

// ***********************************************************
// *************** CONFIG ********************************

// style 0=vertical mount, 1=horizontal (flat mount)
#define STYLE    1

// drawing thickness (0.6mm should be always fine)
#define LTHICK  0.6
// toroid dimensions
#define T_ID   13.3
#define T_OD   23.6
#define T_WIDTH 8.38

// wire details
#define W_THICK   1.626
#define TURNS     18

// wire minimum bend radius
#define MINB   1
// spacing of turns from each other (in mm)
#define GAP    0
// ***********************************************************
// ***********************************************************

void vect_to_cart(double angle, double length, double *x,
double *y);

int
main(void)
{
  int i;
  double wire_outerrad, wire_innerrad, in_circum, out_circum;
  double wound_in_circum, wound_deg, sa;
  double ang_inc, ax, ay, bx, by;

  printf("grid mm\n");
  printf("grid 1\n");
  printf("layer 21\n");
  printf("set wire_bend 2\n");
  printf("wire %.2f\n", LTHICK);

  if (STYLE) // horizontal mount
  {
    // draw circles
    printf("circle (0 0) (%.2f 0)\n", T_ID/2);
    printf("circle (0 0) (%.2f 0)\n", T_OD/2);
  }
  else
  {
    // vertical mount, draw square
    printf("wire (-%.2f %.2f) (%.2f %.2f)\n", T_OD/2,
T_WIDTH/2, T_OD/2, T_WIDTH/2);
    printf("wire (%.2f %.2f) (%.2f -%.2f)\n", T_OD/2,
T_WIDTH/2, T_OD/2, T_WIDTH/2);
    printf("wire (%.2f -%.2f) (-%.2f -%.2f)\n", T_OD/2,
T_WIDTH/2, T_OD/2, T_WIDTH/2);
    printf("wire (-%.2f -%.2f) (-%.2f %.2f)\n", T_OD/2,
T_WIDTH/2, T_OD/2, T_WIDTH/2);

  }

  printf("wire %.2f\n", W_THICK);
  printf("change drill %.2f\n", W_THICK+0.2);
  printf("pad %.2f ROUND\n", W_THICK+2);

  wire_outerrad=(T_OD/2)+MINB+(W_THICK/2);
  wire_innerrad=(T_ID/2)-MINB-(W_THICK/2);
  in_circum=2*PI*wire_innerrad;
  out_circum=2*PI*wire_outerrad;
  wound_in_circum=TURNS*(W_THICK+GAP);
  wound_deg=(wound_in_circum/in_circum)*PI*2;
  //sa=((PI*2)-wound_deg)/2;
  sa=((PI*2)-(((((TURNS-1)*(W_THICK+GAP))-
GAP)/in_circum)*PI*2))/2;
  ang_inc=wound_deg/TURNS;

  for (i=0; i<TURNS; i++)
  {
    vect_to_cart(sa+(ang_inc*i), wire_innerrad, &ax, &ay);
    vect_to_cart(sa+(ang_inc*i), wire_outerrad, &bx, &by);

    if (STYLE) // horizontal
    {
      printf("wire (%.2f %.2f) (%.2f %.2f)\n", ax, ay, bx, by);
      if (i==0)
      {
        printf("pad (%.2f %.2f)\n", ax, ay);
      }
    }
    else // vertical
    {
      if (i==0)
      {
        printf("pad (%.2f %.2f)\n", ax, (T_WIDTH/2)+MINB);
      }

    }


  }
  if (STYLE) // horizontal
  {
    printf("pad (%.2f %.2f)\n", bx, by);
  }
  else // vertical
  {
    printf("pad (%.2f -%.2f)\n", ax, (T_WIDTH/2)+MINB);
  }


  return(0);
}

void
vect_to_cart(double angle, double length, double *x, double *y)
{
  double nx, ny;
  ny=0-cos(angle); nx=sin(angle);
  ny=ny*length; nx=nx*length;

 *y=ny; *x=nx;

}
```

Default layer names and settings summary



Nr		Name
1	🟥	Top
16	🟦	Bottom
17	🟩	Pads
18	🟩	Vias
19	🟨	Unrouted
20	⬜	Dimension
21	⬜	tPlace
22	⬜	bPlace
23	⬜	tOrigins
24	⬜	bOrigins
25	⬜	tNames
26	⬜	bNames
27	⬜	tValues
28	⬜	bValues
29	▨	tStop
30	▧	bStop
31	▨	tCream
32	▧	bCream
33	▨	tFinish
34	▧	bFinish
35	▨	tGlue
36	▧	bGlue
37	⬜	tTest
38	⬜	bTest
39	▩	tKeepout
40	▩	bKeepout
41	▩	tRestrict
42	▩	bRestrict
43	▩	vRestrict
44	⬜	Drills
45	⬜	Holes
46	🟦	Milling
47	⬜	Measures
48	⬜	Document
49	⬜	Reference
51	⬜	tDocu
52	⬜	bDocu

THE END

------ x ------