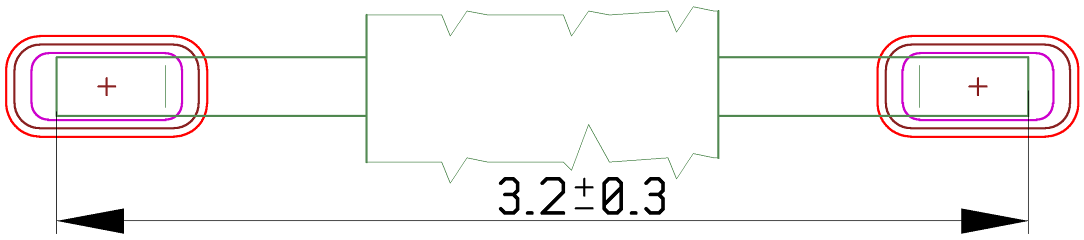

Bedrock-rnd

Version 2025-01-10a

Footprint design manual

Bedrock-rnd - footprint design manual

Bedrock-rnd is a standardized footprint collection for Ringdove EDA.

Ringdove EDA is a Free Open Source Electronic Design Automation software suite.

pcb-rnd is an open source Printed Circuit Board editor, part of Ringdove EDA.

This is the pdf version of the original html based manual. The always up-to-date full version is available online at
http://repo.hu/cgi-bin/pool.cgi?project=pcb-rnd&cmd=show&node=bedrock

Downloads and contacts:

- Ringdove EDA web page: http://www.repo.hu/projects/ringdove
- pcb-rnd web page: http://www.repo.hu/projects/pcb-rnd
- source of this manual: svn://repo.hu/pcb-rnd-aux/trunk/web/pool/bedrock
- contact (email or chat): http://www.repo.hu/projects/ringdove/contact.html

1. Introduction

1.1. About this document

1.1.1. Scope

This document provides all information to design modern footprints with pcb-rnd: overview of footprint design concepts, detailed instructions, references and a check list for manual inspection of footprints (Quality Assurance).

Just like with any other field of engineering, there are many different, often contradicting design goals for footprints. This document focuses on the standardized footprints of the bedrock-rnd footprint library for the Ringdove EDA suite and has arbitrary footprint design choices set up accordingly. While majority of the considerations presented in this document are rather generic and could be used outside of Ringdove EDA, some implementation details are tied to pcb-rnd.

1.1.2. Document style

Text files and code snippets are presented in monospace format:

```
li:pcb-rnd-subcircuit-v6 {
 ha:subc.2{
  ha:attributes {
   footprint = dummy
   refdes = {U1}
  }
 }
}
```

Code 1: Example code style

Bullet lists are:

- foo
- bar

1.2. Why proper footprint is important

For low quantity and/or home built boards footprint details are not that important. When populating the board manually, soldering with an iron, a lot of small mismatches between the footprint and the actual part can be easily worked around.

The more the process is leaning toward serial or mass production, the more such little details start to matter. When using a pick & place machine and reflow soldering, a suboptimal footprint may cause occasional problems, such as misaligned parts or tombstoning. The worse the footprint is, the higher the rate of faulty boards coming off of the production line would be. Thus in most productions beyond the "few pieces of prototypes manually soldered" footprint quality matters a lot and low quality footprints can have a high cost (because of scrap boards or rework required).

1.3. (Non-) solder mask defined footprints

A non-soldermask defined (NSMD) pad has a solder mask opening larger than the copper pad size, leaving a tiny gap exposed between the edges of the copper pad and the edge of the solder resist. A soldermask defined (SMD) pad has a mask opening smaller than the copper pad, which means solder resist opening edge is always on top of the copper pad.

Both NSMD and SMD have their own advantages and disadvantages (see comments by raypcb cadence and macrofab). The general consensus seems to be that NSMD is preferred unless in some special cases, typically with BGA, where pad size is tiny and the part vendor's datasheet explicitly requires SMD.

This document will assume NSMD and recommend deviation only upon explicit request by part vendor datasheet.

1.4. Thru-hole footprints

Most of this document deals with SMD (surface mount) components and footprints, since those are the dominant parts on a modern board. When considerations are extended to thru-hole cases, that is always mentioned explicitly. When SMD/thru-hole is not mentioned, the text is written about SMD.

2. Choosing footprint type and metadata

2.1. Parametric vs. static file

For pcb-rnd there are two types of footprints: *static file* and *parametric*.

A static file footprint is a single data file that fully specifies the geometry of a footprint with all dimensions hardwired. This is most useful in two situations:

- generic footprint for hand soldering; for example the stock sot23 footprint in the footprint library shipped with pcb-rnd
- a special footprint designed for a single part of a specific vendor; for example unique footprints of thick modules, where the same footprint will not ever be used for any other part

A parametric footprint is a script or program that is executed parameters with parameters (on its "argv") and generates a footprint that is then printed to its stdout. The advantage of parametric footprint is that it can handle all presently known and future members of a whole footprint family. A parametric footprint can be used directly from the pcb-rnd GUI (library window), from the "footprint attribute" of a symbol in a schematics or can be executed manually from the shell with its output redirected to a static footprint file.

For high end footprints parametrics can be used to compute optimal pad geometry considering tolerances. This is especially important since different parts coming seemingly in the same footprint may have widely different dimensions, e.g. Texas Instruments has at least two SOT-23 variants whose footprints are incompatible (DBV0003A and DBZ0003A). One way to solve this problem is manually draw a few dozen variants of SOT-23 footprints using footprint recommendations from different vendors/parts. Another way is implementing a parametric SOT-23 footprint which gets main dimensions and tolerances as input and generates the optimal footprint e.g. using the RMS formulas from IPC-7351.

Bedrock prefers the latter and goes for static file footprints only for one-of-its-kind special parts.

2.2. File format

2.2.1. Available footprint file formats

Generally speaking pcb-rnd treats all footprint file formats equally: footprint library entries or stdout of a parametric footprint can be any of file format pcb-rnd supports. However, there are considerations that make some file formats more suitable for shared footprint libs than others.

The *native* file format is lihata. This is the only file format that is guaranteed to support every feature pcb-rnd implements. Thus this is the only format that can store a footprint without compromises, in a lossless way.

Other file formats are called *alien*. These are mainly file formats used by other EDA software. Except for the most notable alien footprint file format which is tEDAx footprint and is a free format designed for data interchange between EDA tools.

2.2.2. Choosing the right format

When designing a new footprint, either parametric or static file, the following points should determine the choice of file format:

- for contributing to bedrock-rnd, the file format must be lihata
- for footprints shared otherwise, it is preferred to use lihata or tEDAx since the plugins of other alien formats are considered less essential and may not be compiled or installed on a random user's system
- lihata is better in describing details so it should be preferred for high quality footprints; tEDAx is easier to understand, read and write so it should be preferred for simplified or low end generic footprints
- a major drawback of the tEDAx footprint block is that it does not convey metadata, while lihata footprints can easily store footprint attributes; in case of tEDAx these can be included in the file as comments, which are then not computer readable; thus for footprints that are part of public libraries lihata is a more suitable format than tEDAx
- when implementing a parametric footprint to be shared, choose a language that is portable and is available; the safest choice is AWK, which is present on any *NIX system by default and the windows binaries of Ringdove EDA installs it too
- another advantage of implementing a parametric footprint in AWK is that it can use common_subc.awk, which is shipped and installed with pcb-rnd; this script lib has functions to convert parameters and write lihata footprint files
- never use the gEDA/PCB file format out of habit; that file format is extremely limited and can not express any fine detail; footprints in this file format are typically low quality ones usable only for hobby boards

2.2.3. Footprint file format specifications

- lihata
  - pcb-rnd data model
  - format low level: syntax
  - format high level: the subcircuit subtree
  - high level struct drawing
  - example
- tEDAx
  - format low level: syntax
  - geometry
  - high level: footprint block (example included)

2.3. Datasheet and reference

The footprint should have metadata pointing to the source material the footprint is based on. This information should be in form of key=value pairs; in lihata format as footprint attributes, in other formats not supporting attributes or tagging as the nearest computer readable feature or as plain comments as last resort.

The following keys should be used:

key	importance	value (purpose of the key)
generator	high	if footprint data is generated by a script, program, parametric footprint, the name::version of the generator should be stored here
datasheet	high	URL pointing to a publicly available datasheet that has the dimensions/drawing of the part or the footprint; it is good practice to get the datasheet mirrored on archive.org's wayback machine and link that instead of a vendor document directly (that is typically removed or renamed long term)
standard	high	a comma separated list of standards the footprint complies to; for referring the current document, use value *bedrock-rnd::version* where version is the version number of the document (see front page).
references	optional	a comma separated list of URLs to secondary resources used for footprint design, such as guides, tips & tricks

2.4. Copyright

The Ringdove convention for licenses on data, such as symbols of footprints, is to use dual licensing, plus generators and scripts may introduce a third level:

- *dist license* controls how the footprint may be distributed as a stand-alone footprint; for static footprint files this means distributing the footprint file (as a stand-alone footprint or as part of a footprint library)
- *use license* controls how the footprint can be embedded in a board file; once the footprint is placed in a board file the relevant license is the *use license*; this doesn't change if the whole board is exported; however if footprint is taken out from the board file and exported into a footprint file, that file is then covered by the *dist license* again
- if the footprint data is generated by a script or program (e.g. a parametric footprint), that script or program may have its own license, the *generator license*; normally the *generator license* controls only how the script/program can be distributed or used, but does not affect the generated footprint data

For *dist license* it is common to choose a permissive free software license that requires the file to remain free. This could be GPL, BSD, MIT or a suitable CC variant. In case *dist license* is GPL and the footprint file is to be distributed as part of a footprint library, GPL may affect the *dist license* on the whole library. However if the GPL'd footprint is used in a board, the board is not affected by the GPL because the *use license* takes precedence upon embedding.

For *use license* the most common choice is **unlimited**, which is interpreted to be equivalent to public domain or CC0. It is not practical to chose any license that requires derivative/combined works to have the same license because that'd affect the board file and could cause footprint license incompatibility.

The *generator license* matters much less as it generally doesn't control the use of footprint data coming from the generator. Most often generators and parametric footprints are coming with a free software license. A notable case is when a parametric footprint is ran directly from within pcb-rnd. There is a clean, well documented API between pcb-rnd and parametric footprints, which makes both ends of the equation replaceable. A parametric footprint does not rely on internals of pcb-rnd and pcb-rnd doesn't make any assumption on the internals of the parametric footprint. The parametric footprint is never linked to pcb-rnd, but is executed as a separate process. Thus the GPL license of pcb-rnd does not affect the generator and the *generator license*, may it be GPL, can not affect pcb-rnd or any other tool that invokes the parametric footprint in a similar manner to pcb-rnd.

Once licenses are chosen, the following metadata (see chapter 2.3) should be set:

key	**importance**	**value (purpose of the key)**
license/dist	high	name or URL of the *dist license*
license/use	high	name or URL of the *use license*
copyright	optional	free form text string containing release year, (C) and author’s name; this field shall not contain any licensing information
author/contact	optional	contact information so that the author can be reached; e.g. email address

Note: license value for license/* is either the short name of a well known license in case it uniquely identifies the license (e.g. GPL2) or is an URL to the license. When distributing footprints in any form, it is best practice to include the full length license file(s) in plain text format next to the footprint file.

2.5. Versioning

For static footprint files it is good practice to include footprint version information in metadata. This helps the user compare board-embedded versions of the footprint to library versions. Version information should be included both for hand drawn footprints and generated footprints in case the generated footprint file is saved and used/distributed as static file footprint.

The format of the version string is arbitrary, but generally should be computer sortable. For bedrock-rnd the preferred format is

`year-month-day`**`suffix`**

where year is a 4 digit year, month is a two digit month, day is a two digit day-of-month and suffix is a single lowercase letter between 'a' and 'z'. For example: 2025-04-21c

In case of parametric footprint the output should not contain footprint version information but the generator field (as specified in chapter 2.3).

Once licenses are chosen, the following metadata (see chapter 2.3) should be set:

key	importance	value (purpose of the key)
version	recommended	user assigned version number
uuid	optional	only if the file format does not have an uuid field natively: a suitably long string of printable ASCII characters which shall be globally unique; only used in static file footprints; helps tracking back footprints in case of name collisions; a suitable tool for generating UUIDs is minuid, installed with sch-rnd at $PREFIX/lib/sch-rnd/minuid; a new version of a static file footprint must have a new uuid value

Note: the lihata file format does have an uid field which is compatible with uuid so lihata footprints should not have an uuid key in metadata. Since the tEDAx file format does not have metadata and comments are not preserved when the footprint is placed on board, uuid in tEDAx can help in footprint file identification but will not help in board vs. footprint comparison.

3.1. Introduction

3.1.1. Terminology

3.1.1.1. Footprint vs. land pattern

Ringdove EDA, pcb-rnd and bedrock-rnd uses the term *footprint* to describe the reusable graphical+metadata block that would normally host a component on a PCB. This document uses *footprint* in this sense and does not use the term *land pattern* outside of this section.

The IPC-7351 standard uses the term *land pattern* for the same concept and uses *footprint* for the surface where the component actually touches copper pads, where solder joint is made.

From fabrication point of view it is often defined as *footprint* being the actual physical surface where the component touches the board (so it is the geometry of the component) while *land pattern* is the geometry of the copper pads on the PCB. In this interpretation a *land pattern* is always bigger than the corresponding *footprint*. A single *land pattern* may be uses with slightly different parts (*footprints*). Example definitions: madpcb , a PCB HERO employee, protoexpress.

Some fabs may have slightly different interpretations: raypcb

3.1.1.2. Solder joint details (SMD)

The terminology for the solder joint of an Surface Mount Device (SMD) is modeled after the anatomy of a leg, as shown on Fig. 3.1.

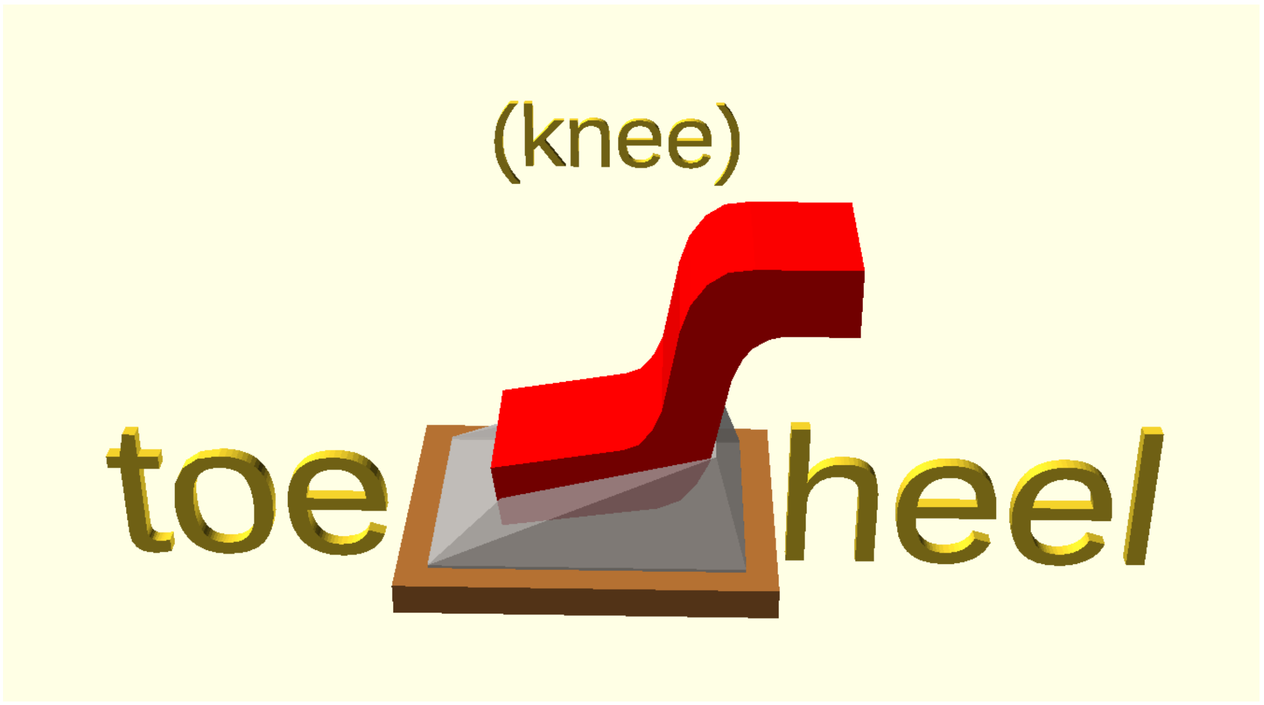

Figure 3.1. gull-wing lead (SMD) soldered to a copper pad; the grey blob is the solder joint

The two ends of the solder cone along the short edge of the pad are called *toe* (at lead end) and *heel*. The solder cone along the longer edges of the pad are called *side*.

In footprint design for SMD components the basis for determining the dimensions of the copper pad largely depends on the expected shape of the solder cone. Taken vertical cross sections of the above image, the solder cone is somewhat triangular with different slope angle on the toe, heel and side parts. Taking a series of horizontal cross sections of the solder cone, from top to bottom, the cross section area is constantly increasing until we reach the copper pad.

This is just a crude approximation: in reality the outer walls of the solder cone are not flat planes but curved. But for footprint design that doesn't matter much, we are interested only in the top and bottom horizontal cross sections. The top cross section's area is basically the area of the of the part's lead (pin) that would touch the PCB when placed on top of the PCB, while the bottom cross section is the area of the pad. Pad area is almost always larger than lead/pin area, to get a reliable and mechanically stable solder joint.

3.1.2. Tolerances

The archetypical example of having to deal with tolerances is when there are two pads (e.g. in 1206) or two rows of pads (e.g. in SOIC) in the footprint. From the component's point of view this means two rows of leads/pins. Normally the pad would be as small as the bottom side of the lead/pin touching the PCB plus some extra area for the solder cone (as described in 3.1.1). However there is a certain manufacturing tolerances, which means actual components will come in the smallest possible size (LMC, Least Material Condition) and largest possible size (MMC, Most Material Condition), from the same vendor, with the same part number. Pad dimensions should be determined so that the footprint can host both the LMC and MMC case (as shown on Figure 3.2). In other words the footprint always works for the given component within the tolerance specified by the vendor.

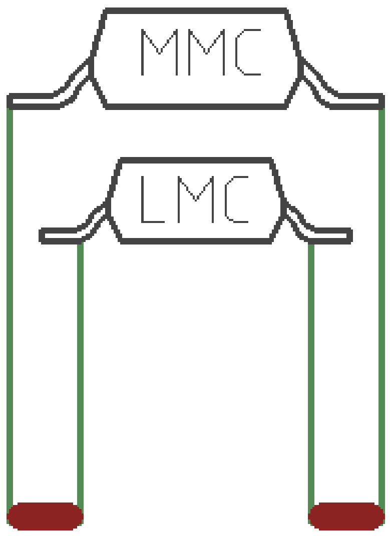

Figure 3.2. the same gull-wing leaded component ended up at MMC and LMC size determining minimum pad length; pad drawn with red

Furthermore there are other tolerances to consider, such as the tolerance of PCB manufacturing or the tolerance of component placement.

The naive solution to the problem would be computing the absolute worst case extremes: when the largest (or smallest) part is placed furthest away from the originally planned position on a board printed with the worst displacement of the pad possible. This would obviously result in a footprint that really always works, but at a cost of very large pads. The industry settled with a different method for accumulating tolerances: the RMS (Root Mean Square).

When a value A has a tolerance, that means there's a minimal (LMC) and a maximal (MMC) value A can take. Tolerance of A is then:
$A_{tol} = A_{max} - A_{min}$

With RMS if there are 3 different source of errors specified by tolerance, A_{tol}, B_{tol} and C_{tol}, the final tolerance value to use is
$sqrt(A_{tol} * A_{tol} + B_{tol} * B_{tol} + C_{tol} * C_{tol})$.

Implementation details are described in section 3.4.1.

3.1.3. Density and soldering process

Different boards may have different density goals. For example on a cell phone motherboard components need to be packed very tightly even if that means more expensive fab processes, while on cheap, throw-away consumer electronics where PCB size is not a limiting factor it’s better to go for larger pads and larger footprints in general to reduce costs and risks.

In footprint design this generic density goal is indicated by ’A’, ’B’ or ’C’. ’A’ is the least dense (consumer) board, ’C’ is the tightest, most dense board. (Some parts of the industry also call this ’L’, ’N’ and ’M’, but this document and bedrock-rnd uses the A..C convention exclusively).

A board designed for manual soldering should normally go for ’A’ footprints. For reflow process ’B’ or ’C’ is better. Wave soldering may be close to ’A’ with some more tweaking specific to the process.

Unless otherwise mentioned, this document assumes ’B’ and that is the recommendation for normal high end footprints for the general audience. Rationale: for hand soldering much simpler, generic, low end footprints can be used and those won’t require most of the advanced considerations of this doc (as the user won’t even use the past layer for example). For wave soldering or critically dense ’C’ boards the designer will have to manually deal with each and every footprint and match them up with the specific process available, well beyond the scope of this document. The following attribute/metadata key should be used:

key	**importance**	**value (purpose of the key)**
density	recommended	a single letter A, B or C

3.2. Origin, centroid, orientation

3.2.1. centroid

Every footprint has an centroid (origin). In pcb-rnd this is the 0;0 coordinate in the footprint-relative coordinate system and pcb-rnd also draws a small red diamond at that point.

The centroid has two purposes:

- by default it is the pick & place grab point
- on the GUI this is the point by which the subcircuit is grabbed when initially taken from the library

Because of the former, it is a good idea to have the centroid at the center of the component body, unless the component body has a very uneven top side or the center of gravity is far off from this point.

Note: due to the GUI drag point aspect, some older footprints used to have the centroid at pin 1. This may seem like a good idea when placing the footprint on a grid, having pin 1 aligned to the grid. However this is absolutely not necessary: after a random placement, it’s very easy to get the pcb-rnd GUI to snap to pin 1 or whichever other pin and the subcircuit can be grabbed by that snapped off-grid point and will align to the grid once placed.

In the rare case when the centroid can not be matched with the pick & place center, a dedicated pnp-origin object shall be added.

3.2.2. orientation

For this chapter we are looking at the component (or the board with the footprint on the top side) from the top.

There are conventions for the standard footprint orientation, which means 0 degree rotated (both in pcb-rnd and for an assembly fab house). The generic rule is to have pin 1 on the top left side, with pin 2 under it. Pin numbering should go around like with a DIP package. This generic rule works well for almost all footprints. For example a SOT-23-3 would have pin 1 on top left, pin 2 on bottom left and pin 3 on middle right. QFN/QFP footprints should have pin 1 as the top pin on the left side row of pins.

A notable exception is PLCC, which has pin 1 on middle top for historical reasons.

Another exception is the ’bottom terminated parts’, like BGA, where there is often a matrix of pads underneath the component. These parts typically have custom pad naming scheme because of the large number of pads.

3.3. Padstack design

3.3.1. Avoid overlaps

Avoid using overlapping padstacks, especially for constructing complex pads.

For example for mesh paste pattern over a center pad (e.g. in QFN) use a single center pad padstack and use plain rectangles (not padstacks) on the paste layer. The rectangle objects could be associated with the terminal by setting their term attribute (heavy terminal concept).

In such a center pad if vias are also required, make sure the via does not have copper, paste and mask on the same layer as the centerpad rectangle is on (top layer normally).

gEDA/PCB used to have many limitations which forced the user to use multiple pad objects; these problems have been resolved in pcb-rnd for many years by now. If a custom shaped padstack is required, just use a polygon shape If complex paste or mask patterns are required, don't do them as padstacks but simply draw them as layer objects (both paste and mask layers are editable).

3.3.2. Rely on padstack rotation

Many footprints have the same pads on two or four sides, e.g. SOIC or QFN. Draw the padstack only once and use the same padstack for each side with the appropriate n*90 rotation. Use the least number of different padstacks possible.

3.3.3. Padstack origin

Most padstacks will be circular, square-like or rectangular. In the former two cases it's best to have the padstack origin in the center of the padstack. In the rectangular case, the origin shall be in the center along the shorter edge; along the longer edge pick a meaningful internal point on the pad (e.g. center, toe or heel).

This rule is to help the GUI to snap while routing traces.

3.3.4. Terminal IDs

All terminals should have a terminal ID; when the datasheet has pin/pad numbering, that shall be used. When the datasheet does not have an ID for a terminal that is expected to be connected ever, assign a non-colliding terminal ID:

- for example connector's chassis terminals could be called 0 or chassis
- if there are multiple terminals for the same purpose
  - if they are equivalent (e.g. DSUB connector chassis) name them the same; that assumes internal connection and it will be enough to connect one

- if they each need to be connected, e.g. multiple ground pins/tabs, have different names for them and d not use intconn; this way the schematics will need to connect them all
- if the difference between the terminals is significant and there is internal connection the user will rely on, use different terminal IDs and intconn; for example a 0 ohm nonetlist jumper resistor would have terminal IDs 1 and 2 and intconn in between

3.4. Copper (and holes)

In reflow soldering using the correct pad size is very important. A pad too large can cause:

- tombstoning
- the gap between adjacent pads become smaller which increases risk of solder bridging
- excess solder paste around a much smaller component lead may form solder beads
- floating parts: component misalignment (component is not centered by surface tension of molten solder)

while a pad too small can cause:

- (thru-hole) solder wicking
- (thru-hole) drill breaking the ring
- insufficient solder joint
- component termination not reaching pad or not producing enough overlap with pad causing tombstoning or missing solder joint on one side

Some other layers (mask, paste) are partly or fully based on the copper pad shape and dimensions.

3.4.1. SMD copper pad size from component lead/pin size

For static file footprints copper pad sizes should be used from the datasheet’s footprint drawing if it is present. In that case skip 3.4.1.

If footprint dimensions are not available, or a parametric footprint is created for a family of toleranced components, the method described in this section should be used to determine copper pad dimensions.

The most usual case, as mentioned in 3.1, is when there are two rows of leads/pins on opposing sides, as shown on a gull-wing package (e.g. SOIC) on Figure 3.4. below.

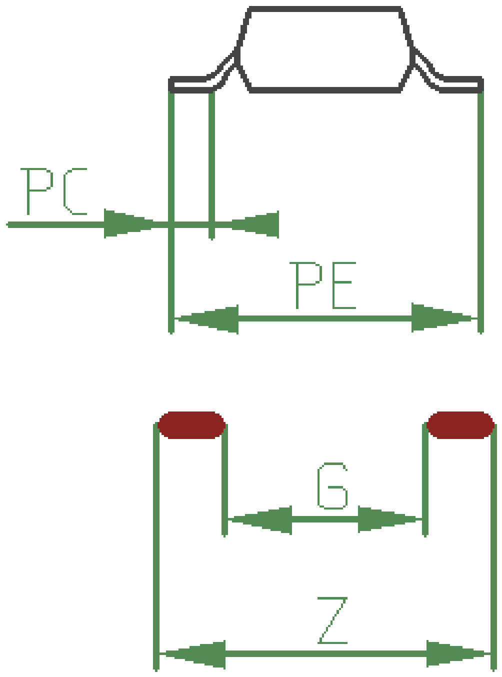

Figure 3.4. gull-wing pad length with tolerances accumulated

Input is PE and PC, which are typically specified in the datasheet as a dimension with tolerance. PE is the extent measured between ends of leads and PC is the constant length (flat part of the lead where it contacts the PCB). Expected output is Z and G, which are the extent of and gap between the pads. All dimensions are in mm and all input dimensions have tolerances. Output should be determined so that both LMC and MMC parts fit.

The minimum, maximum and tolerance values for an input, e.g. PE, is defined as:

- PE_{min} smallest allowed value of PE (LMC)
- PE_{max} largest allowed value of PE (MMC)
- $PE_{tol} = PE_{max} - PE_{min}$
- $PE_{nominal} = (PE_{min} + PE_{max})/2$

Some datasheets specify LMC and MMC, others specify LMC and tolerance, yet others specify nominal and tol. The other two, unspecified values are always easy to calculate.

The outer extent of the copper pad is the maximum value of Z:

```
Z_max = PE_min + 2*J_T + sqrt(PE_tol*PE_tol + F_tol*F_tol + P_tol*P_tol)
```

Code 3.4. computing pad extent

Where J_T is the bottom size of the solder joint at the toe, F_{tol} is PCB fabrication tolerance (typically 0.1mm), P_{tol} is the tolerance of component placement (typically 0.05mm). The value of J_T is added twice because Z is a span from end to end containing two solder joints. The value depends on lead shape and size and board density and is typically between 0.15mm and 0.8mm. The result, Z, may be rounded to the nearest 0.05mm or 0.1mm.

Gap size G is computed in a similar manner:

```
Stol_RMS = sqrt(PE_tol*PE_tol + 2*PC_tol*PC_tol)
PG_max = PE_max - 2*PC_min
G_min = PG_max - 2*J_H - sqrt(Stol_RMS*Stol_RMS + F_tol*F_tol + P_tol*P_tol)
```

Code 3.5. computing gap between pads

Where J_H is the bottom size of the solder joint at the heel (used twice because of the two pads included in the gap), typical values ranging from 0.15mm to 0.8mm. $Stol_{RMS}$ is the total pad length tolerance considering the tolerances of input pin extent and input contact length. PG_{max} is the worst case pin gap on the component. The meaning of J_H and J_T are shown on Figure 3.5.

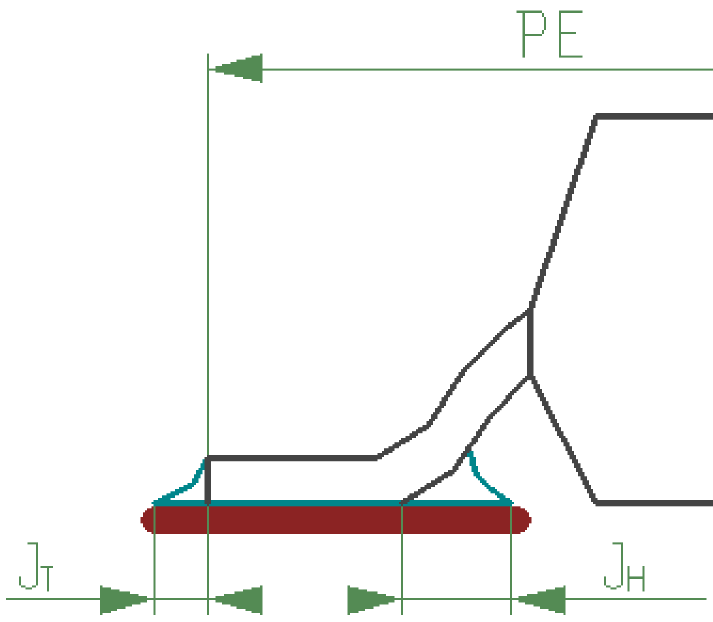

Figure 3.5. PE, JT and JH on a gull-wing lead solder

Finally a very similar computation is done on a perpendicular plane to find the height of the pad. The input for this is the width of a lead/pin, PT and the expected solder joint side size, J_S, typical values ranging from 0mm to 0.3mm.

```
Xmax = PT_min + 2*J_S + sqrt(PT_tol*PT_tol + F_tol*F_tol + P_tol*P_tol)
```

Code 3.6. computing pad height

3.4.2. pad shape

SMD pads are typically rectangular. It is preferable to have rounded corners with a radius of about 25% of the shorter edge of the rectangle but not more than 0.25mm. There are multiple reasons for the rounding:

- corners are asymmetrically far away from where the solder joint happens
- the solder may not even flow into the corners during reflow, so preserving sharp corners won’t add to the working area
- the paste layer should have rounded corners for even stronger reasons, having a similar copper pad shape looks better

Note: since the area lost in corner rounding wouldn't have participated much in the solder joint, the basic dimensions of the pad shall not be increased as a compensation for rounding.

3.4.3. vias and center pad

The most common reason for a footprint to contain vias is center pad heat bridging (e.g. on qfn or qfp parts). In such case it is best to create a large rectangular padstack for the center pad, with no paste and another via padstack with no top side copper, paste or mask. Top side copper should be provided by the rectangular pad only. (The paste layer will need to have a mesh pattern in this setup; this is discussed later).

If vias are added for other reasons, e.g. fanout, consider placing the via away from exposed copper pads. If the via absolutely needs to be placed on the pad, paste patterns and/or via plugging needs to be considered.

3.4.4. plated holes (vias, thru-hole pins)

The footprint shall always specify finished hole diameters, not tool diameter.

3.4.4.1. Ring size and circular drill

Minimum concentric copper diameter around a circular hole shall be:

$D_{coppermin} = D_{hmax} + 2 * R + T$

Where (size recommendations assume density 'B') for external copper layers:

- $D_{coppermin}$: minimum copper pad diameter
- D_{hmax}: maximum diameter of the finished (plated) hole
- R: minimum ring width; if not specified in the datasheet: should be 0.2mm for hole size of 1mm; generally speaking R should be $D_{hmax}/5$ in this range, but should never be smaller than 0.15mm; for large holes of heavy components, over 2mm, copper diameter can be increased more
- T: accumulated tolerances of fabrication; should be 0.2mm

On internal copper layers R and T could be divided by 2, but R must be at least 0.1mm; use smaller internal copper only when pitch is smaller than 0.127mm.

D_{hmax} should be chosen to be 0.35mm larger than the maximum pin diameter and if not already round, rounded up to the next 0.1mm value. For non-circular pins diameter is measured across the diagonal.

For example a typical DIP package would have a 0.6mm max pin diameter (diagonal), which would result in an 1mm finished hole (0.95+rounding); copper pad diameter: $D_{coppermin} = 1.0 + 2*0.2 + 0.2 = 1.6mm$

The copper ring for pin 1 shall be square (polygon shape in the padstack).

3.4.4.2. Slots for tabs

Generally the rules of 3.4.4.1 apply, but D_{hmax} is at least 0.5mm for plated and 1mm for unplated. Slot mechanical layer shall be drawn with round cap line in a padstack.

3.4.4.3. Unplated holes

Some components (even SMD ones) may have plastic pins or tabs for positioning or increasing mechanical stability. The padstack of these holes must be marked unplated and the padstack shall not have any copper or paste shape but should have a mask opening larger than the hole by at least 0.1mm (so that the hole is not tented).

3.5. Mask

3.5.1. Shape and size

Unless the datasheet of the component explicitly requires, go for NSMD (non-soldermask-defined) padstacks, see chapter 1.3. This means the solder mask opening shall be slightly larger than the copper pad. Normally the shape is the same and the gap should be between 0 and 0.1mm (according to IPC). Some PCB fabs, like jlcpcb, have their own minimum gap (0.038mm as of early 2025). This is probably to compensate for mask placement tolerance (so that a slightly misplaced mask does not reduce copper pad exposure).

Recommended gap size is 0.075mm; if the footprint is designed for the 'C' density level, recommended gap size is 0.045mm.

3.5.2. Mask above row of pads

If there is a row of pads, e.g. in case of SOIC or TQFP, each padstack shall have its own mask shape. With reasonable pitches there will be enough mask left in between pads to meet PCB fab minimal requirements (0.1mm at jlcpcb in early 2025). However if the mask bridge between pads is too small because of the pitch being too small, it is better to remove mask from over the whole row. This can be done in two ways:

- keeping mask shape on the padstacks but making them slightly wider so adjacent padstack masks join
- not having mask shape in padstacks but having rectangles drawn over the rows of pads on the mask layer within the footprint

The two methods shall not be mixed.

3.6. Paste

3.6.1. Paste size and shape

Paste size should always be without sharp corners, especially for small area pads. Rounded corners improve paste release and reduces the need of stencil cleaning between paste applications.

The area of paste (together with the thickness of the stencil sheet) determines the volume of solder paste dispensed on the pad. Too little and the solder joint will not be reliable, too much and the reflow process can produce free wandering solder beads or a smaller SMD part can produce tombstoning. IPC7525a has details on stencil design. The rule of thumb for size is:

- generally use the same or slightly smaller size as the copper pad; for example for rectangular pads of gull-wing:
  - with leaded solder paste decrease width by 0.03..0.08mm, decrease length by 0.05..0.13mm
  - with lead-free solder paste decrease width by 0.254mm and preserve length

3.6.2. Advanced shapes

For 2 terminal passives that are 0603 or larger it is recommended to remove some paste in the following pattern, to reduce the risk of forming solder beads during reflow (Figure 3.6).

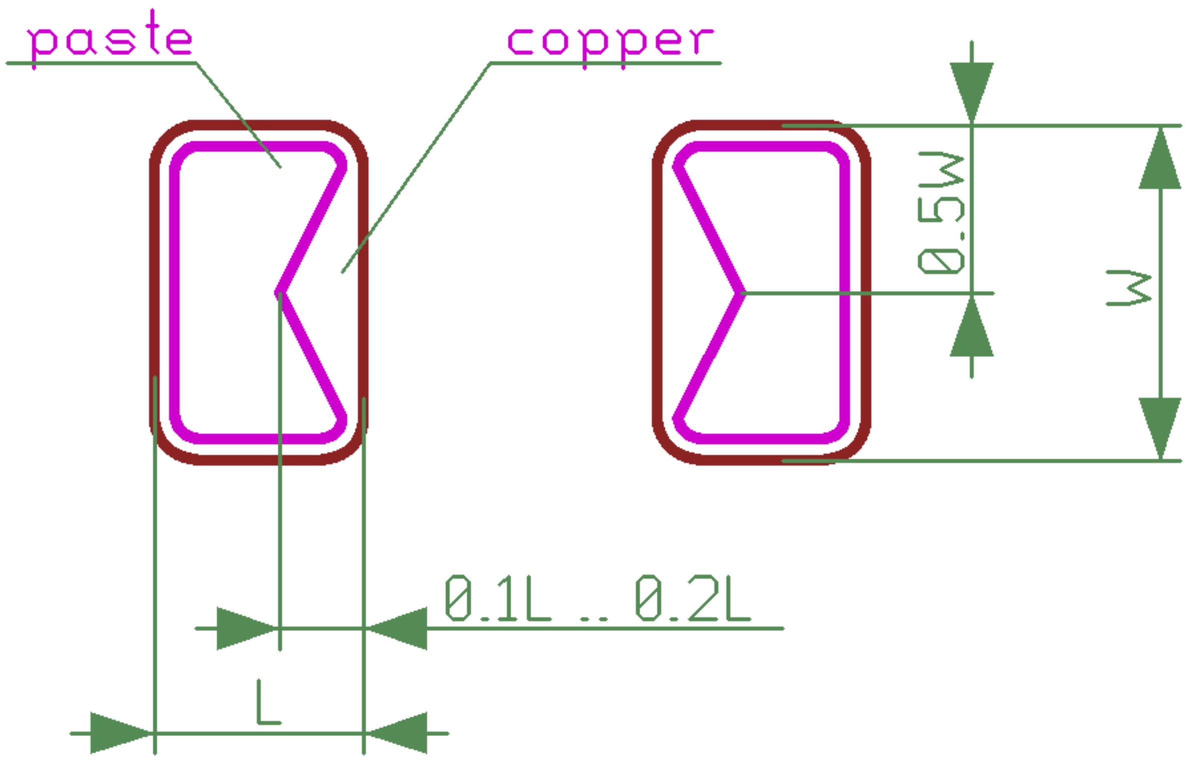

Figure 3.6. inverted home plate paste pattern for chip resistors/capacitors

Cylinder cap SMD like melf/minimelf requires C-shaped paste.

These should be done as polygon paste shapes in the same padstack.

There are a bunch of other shapes in use.

Bob Willis shared an excellent video demonstrating how stencil pattern avoids beading.

3.6.3. Paste patterns

Center pads of footprints like QFN, QFP, some SOIC, etc, are often used for heat sink purpose. This requires vias to be added and such vias could easily suck in solder paste (we normally do not assume plugged/filled vias). Another problem is having too much paste on a large pad, especially in one large solid area, is higher solder dome during reflow on which the component would float causing greater distance to pcb on the signal pads. To avoid these problems a mesh pattern of smaller apertures should be used, as shown on Figure 3.7.

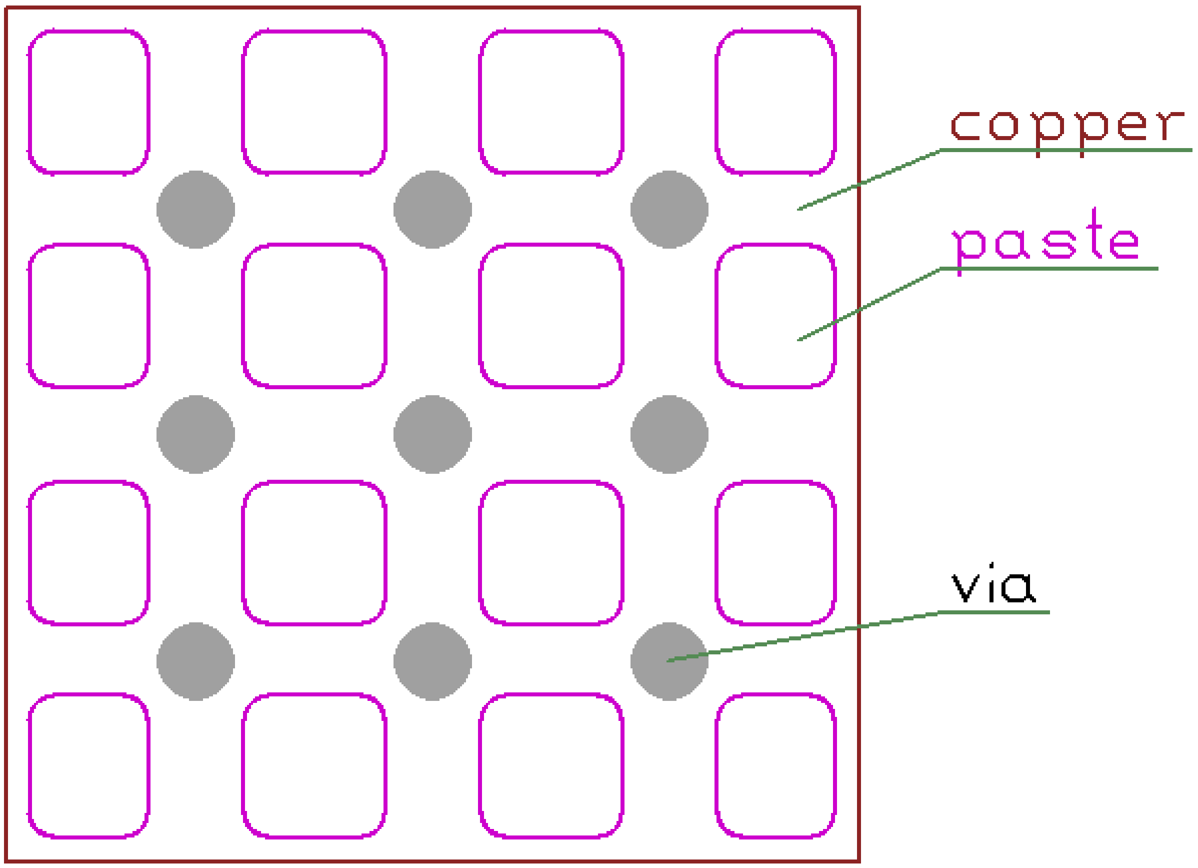

Figure 3.7. center pad mesh pattern on the paste layer

This shall be done using separate round corner rectangle polygons on the paste layer (and not using padstacks), while the center pad padstack itself shall not have a paste shape. Paste should avoid vias.

Total area of the paste should be between 50% and 80% of the area of center pad copper.

As a second line defense against via sucking up solder paste, it's also recommended, probably by IPC-7093A to draw the same mask pattern as the paste pattern. This way the paste+mask (over the larger copper pad) is Solder Mask Defined. The mask grid between the paste meshes would tent vias and potentially block adjacent paste areas to merge during reflow.

To avoid problems with minimum mask width, make the gap between paste meshes at least 0.3mm (NXP recommends 0.4mm in section 4.2.2.2). The mask, fully covering vias, should extend at least 0.15mm beyond via drilled hole. For example if the via hole is 0.3mm, the mask (and no-paste) diameter over the via should be of diameter 0.6mm or larger.

According to NXP a good choice for viagrid would be 0.3mm vias on a 1.2mm pitch regular grid.

Paste pattern meshes should also have a small chamfer or rounding to avoid sharp corner and potentially easy paste release.

Example centerpad-only footprint, with 0.3mm via hole, 0.3mm mesh separation, SMD (mask+paste mesh) over a large centerpad.

3.6.4. Paste overhang

In some special cases the paste shape may go over the edge of the copper pad shape. The excess paste will be sucked up by surface tension during reflow. The reason for using paste pattern slightly larger than the copper pad is typically related to stencil printing technology:

- increase volume of paste added to small area pad
- improve paste release when dealing with very small apertures
- improve paste release when the wall of the stencil is rough

Avoid overhang unless there is a strong, documented reason to use it.

3.7. Assembly

The assembly drawing is extremely useful for the designer in manual inspection while considering how the board would be built or reworked. Each footprint shall have a proper assembly drawing, typically on the top-assy layer.

The assy drawing should be done using thin lines (recommended: 0.05mm thickness with occasionally 0.025mm thick lines e.g. for indicating lead bends). It should show an orthogonal top view of the part with only the main features present. The drawing should include pins/leads. The drawing should include pin 1 marking (if applicable) but other marking, especially engraved text shall be omitted.

The assy drawing is normally done at nominal dimensions.

For components that need to be located relative to the board edge the assy drawing may have a line indicating the expected position of the board edge (without textual comment/explanation).

The assy drawing shall not have anything else, especially not indication of keepout areas or generic documentation.

3.8. Silk

The silk layer shall have:

- the refdes (as a floater text object)
- a dot indicating pin 1 next to the pin; this shall be a zero length line of thickness 0.3mm
- 0.2mm thick lines indicating placement

Use placement indication graphics sparingly: don't draw full boxes. Either draw two lines on the no-pin sides (e.g. in case of SOT-23-5 or SOIC) or right angle corners of two lines each (e.g. SOT-23-3). The purpose of the silk graphics is to give a slight hint on where a part is located - it is not the assembly layer, it does not need to fully specify the component's shape or bounding box.

The pin 1 indication and refdes text both must be visible after the component is soldered on the board. Placement graphics may be partially hidden by the component. Placement graphics must not extend beyond the courtyard area.

3.8.1. Refdes text

The refdes text must be a text object with the dyntext and floater flags set. The initial position of the refdes text should be outside of the courtyard. Refdes text size should be 100% with pcb-rnd default font, resulting an 'M' height of about 1.2mm.

3.9. Courtyard

The courtyard layer shall host (normally a single) polygon. When there are multiple polygons on the layer, they shall not overlap or touch. Any part of the component body, pins/leads and exposed copper pads shall be covered by polygon(s). Input dimensions should be considered at their maximum (MMC). The courtyard shall extend beyond its minimal size by a certain safety margin in every direction. The recommended safety margin depends on target density:

- 0.1mm for ’C’
- 0.25mm for ’B’ (recommended)
- 0.5mm for ’A’

3.10. Keepout

A keepout layer shall host (normally a single) polygon. When there are multiple polygons on the layer, they shall not overlap or touch.

Different keepout types are represented by different keepout layers. In footprints the most commonly used would be ko@top-copper to avoid top side copper directly under some sensitive parts of a component. However, it is important to note that high end footprints shall assume solder mask present which means bottom side copper features on the component alone does not warrant a ko@top-copper since the solder mask would insulate. Better reasons are radio frequency (e.g. antenna) or magnetic (e.g. coil in a DC/DC brick).

Do not use ko@ to indicate mechanical keepout, use the courtyard layer for that.

Having a keepout layer in a footprint is rare. Do not improvise keepout layers: draw them only if the datasheet requested it.

3.11. Adhesive

The adhesive layers is used to draw where glue should be dispensed under component body in case the component needs to be fixed to the board before soldering. This may be the case for larger SMD components on the bottom side of an SMD board that is populated on both sides or for wave soldered SMD boards.

The main problem with the adhesive graphics is that all aspects and details of gluing is highly process dependent. It is very unlikely the PCB designer or the footprint designer could figure when to use the glue and what the right amount of glue is or even how the glue is dispensed (using stencil or nozzle).

Thus drawing the adhesive layer is optional. When drawn for a general purpose footprint, it should be done with either as a thin line which indicates the centerline or multiple small dots (zero length lines of reasonably small but visible diameter). The intention is that the fab house could import this layer as a hint on where gluable components are on the board and use relatively simple transformations (increasing line thickness) to tune glue for their actual process.

3.12. 3D models

3D component models are optional and are not stored directly in the footprint file but as a reference to an external model file via footprint attributes. At the moment the best option with pcb-rnd is OpenSCAD.

Since any 3D model use requires footprint attributes, the tEDAx footprint format should be avoided and the lihata footprint format used when a 3D model is supplied.

If the footprint is parametric, the joined OpenSCAD script should have all the relevant parameters of the footprint also implemented and the parametric footprint script shall generate the appropriate *openscad-param* attribute in the footprint to get the 3D model match the footprint.

The 3D model should come with the same use-license and dist-license as the footprint. The OpenSCAD exporter in pcb-rnd typically copies the component model into the output so use-license that would "contaminate" the output (e.g. GPL) should be avoided.

A. Appendix

A.1. Example

A.1.1 Example footprint

A three pin SOT23 lihata footprint file has been generated using this guide, IPC7351 and the component drawing of a generic SOT23 datasheet from NXP. Copper pad sizes are computed using the method described in section 3.4.1.

The resulting footprint is very similar to the vendor recommended footprint with dimensions matching within the tenth of mms.

The generated footprint has all major features listed in this guide and passes all test of the checklist in the appendix A.3.

A.1.2 Template board for manual footprint drawing

A pcb-rnd template board is available for manual footprint drawing:

- load the template, draw footprint graphics then select and convert to subcricuit
- the template contains all layers normally used in a footprint
- the primary side of the footprint should be top side of the board
    - smd footprints should use only the top side
    - thru-hole footprints should normally use the bottom side only in padstacks, except for special components that really span physically to both sides
    - internal copper is relevant only in thru-hole padstacks
- keepout layers other than courtyard are not created; right click on an existing group in the layer selector to create a new keepout layer group
- predefined routing styles and objects:
    - use routing style *silk-placement* for drawing placement graphics on top silk
    - there is an 1*1 mm corner right angle drawn with *silk-placement*, often just copying and rotating this should be enough for all placement graphics of a footprint
    - there is a 0.3mm diameter circle (zero length line) placed on top silk for pin 1 mark; select it before moving it
    - use routing style *assy-normal* for drawing the assembly layer body and pin/lead outline
    - use routing style *assy-thin* for indicating pin/lead bends and other secondary (non-contour) features
    - there is a 0.25mm diameter circle (zero length line) placed on top assembly layer for pin 1 mark; select it before moving it
- make sure all layers are visible before selecting everything for the subcircuit conversion
- when converting selected object to subcircuit use the menu - that will run an extra step of

querying for the centroid
- no need to remove unused layers and layer groups; subcircuit conversion will include only the relevant layers

A.1.3 Example centerpad paste pattern

Example centerpad-only footprint, with 0.3mm via hole, 0.3mm mesh separation, SMD (mask+paste mesh) over a large centerpad. This demonstrates paste pattern considerations discussed in 3.6.3.

A.2. Standards, resources, links

Below is a collection of external documents covering the topic of footprint design.

A.2.1. Standards

- IPC7351 - Generic Requirements for Surface Mount Design and Land Pattern Standard
- IPC7525a - Stencil Design Guidelines
- IPC/EIA J-STD-001C - Requirements for Soldered Electrical and Electronic Assemblies
- IPC-SM-782 - Surface Mount Design and Land Pattern Standard (much older than IPC7351 but has per footprint class recommended dimensions)
- IPC753x with package dimension tables (Russian): IEC-61188-5-2 IEC-61188-5-3 IEC-61188-5-4 IEC-61188-5-5 IEC-61188-5-6 IEC-61188-5-8
- IPC-CM-770E - Guidelines for Printed Board Component Mounting (for thru-hole considerations)
- IPC2221A - Generic Standard on Printed Board Design (for thru-hole considerations in section 8.3. and 9.); older revision
- IPC2222 - Sectional Design Standard for Rigid Organic Printed Boards (for thru-hole considerations)

A.2.2. Resources

- lihata file format specification:
  - pcb-rnd data model
  - format low level: syntax
  - format high level
- tEDAx file format specification

A.2.3. External links

- Principles of land pattern design - presentation on IPC7351
- Mentor Graphics' app note on package names and zero orientation for each (based on IPC7351b) ; same from IPC/IEC
- List of IPC-compliant package names (syntax)
- PCBWay's analysis on tombstoning
- TechSparks Comprehensive Guide to PCB Stencils
- stencil design basics (paste layer) with dimensions for common SMD parts
- detailed analysis on tombstoning
- TI application note on QFN footprints

A.2.4. Other footprint design guides

- Sierra Circuits proto express (SMD, Thru-hole, all in inches)

A.3. QA checklist

This quality assurance checklist can be used to verify the compliance of a footprint to the bedrock-rnd guidelines. There are two levels of compliance: to get included in the bedrock-rnd library all checks must be passed; to have a high quality footprint that complies with this document but does not get included in bedrock-rnd, some checks can be failed or passed with alternative approach. These alternatives are written in *italic*, often just stating that a check is (*optional*).

Each row starts with an [id] that identifies the row. This is used instead of numbering so that if new checks are inserted later the ids will not change.

While the table blow is designed for manual verification with a checkbox provided in the last column, most of these test can be automated.

Type, format	**ok**
[type] parametric for family, static file for special (*optional*)	☐
[awk] parametric footprint is written in awk and is portable (*optional*)	☐
[name] reasonable footprint name, no collision with existing bedrock-rnd footprint name (*optional*)	☐
[format] footprint is in lihata subcircuit >=v6 format (*or in tEDAx*)	☐

Metadata	**ok**
[generator] has the generator attribute if generated	☐
[datasheet] has the datasheet attribute	☐
[ds-archive] datasheet URL points to archive.org	☐
[standard] has the standard attribute and lists bedrock-rnd with the correct ::version	☐
[license/dist1] has a license/dist attribute with a known license or URL	☐
[license/dist2] dist license is acceptable for bedrock-rnd (*optional*)	☐
[license/use1] has a license/use attribute with a known license or URL	☐
[license/use2] use license is "unlimited" for bedrock-rnd (*optional*)	☐
[copyright] has a copyright attribute with the name of the author and year (*optional*)	☐
[version1] has a version attribute (*optional*)	☐
[version2] version format is yyyy-mm-ddS (*optional*)	☐
[uuid1] has a proper UUID; uid in lihata or manually generated uuid in other formats (*optional*)	☐
[uuid2] UUID format is minuid (*optional*)	☐
[density] has a density attribute that is valid and matches the footprint (*optional*)	☐

Centroid/orientation	**ok**
[centroid] centroid placement is correct for pnp	☐
[orientation] standard orientation, pin 1 location	☐

Padstacks	ok
[pstk-olap] no unreasonable overlaps	☐
[pstk-rot] no excess duplicate transformed padstacks; padstacks are rotated properly where needed	☐
[pstk-origin] proper padstack origin (along short vs. long side)	☐
[termid] terminal IDs are set	☐
[intconn] intconn is not used when not needed or used exactly as required when needed	☐

Copper	ok
[cop-dim1] copper dimensions are either copied from datasheet's recommended footprint drawing or derived from the component drawing; for footprint-family it must be the latter	☐
[cop-dim2] if component-derived sizes: tolerances are properly accounted for	☐
[cop-round] sharp corner pads except for central pads, have rounding that is not larger than the radius recommended in 3.4.2; except for center pads or when there is a strong, documented reason for deviation	☐
[vias] proper via placement (if there are vias)	☐

Hole/slot (if applicable)	ok
[hole-fin] holes are specified at finished diameter	☐
[hole-unplt] mounting/positional/nosolder holes are not marked plated	☐
[hole-untent] non-plated holes have mask opening to avoid tenting (see 3.4.4.3)	☐
[hole-ringdia] minimum copper diameter (see 3.4.4.1)	☐
[hole-pin1] pin 1 has square ring	☐
[slot-pstk] slot is in a padstack and has round cap line shape	☐
[slot-dia] slot respects minimum diameter (see 3.4.4.2)	☐

Mask	ok
[NSMD] uses NSMD or if not, the reason is strong and documented	☐
[mask-shape] mask shape is the same as copper pad shape; if not, there is a strong, well documented reason	☐
[mask-gap] for NSMD the mask gap is as recommended in 3.5.1 (*optional*)	☐
[mask-bridge] there is no narrow mask bridge (minimum: 0.1mm)	☐
[mask-row] if there is a mask rectangle over a row of padstacks those padstacks do not have mask shape	☐

Paste	ok
[paste-shape] paste shape is reasonable for the pad shape and contact lead shape	☐
[paste-rounding] no sharp corner: have rounding that is not larger than the radius recommended in 3.4.2; except for center pads or when there is a strong, documented reason for deviation	☐
[paste-size] equal to or slightly smaller than the copper pad; alternatively strong, documented reason (e.g. overhang)	☐
[paste-tune] flat bottom large pad parts (e.g. passives or tabs of power components) have the recommended paste removal on the inner side, see 3.6.2 (*optional*)	☐
[paste-pattern] large pads (e.g. center pads) have paste pattern instead of a single large rectangle	☐
[paste-via] no paste over unplugged via	☐

Assembly layer	ok
[assy-thick] line thicknesses are as recommended: 0.05mm and 0.025mm (*optional*)	☐
[assy-view] orthogonal top view	☐
[assy-body] has a reasonable representation of the component body	☐
[assy-pins] has a reasonable representation of pins/leads	☐
[assy-pin1] has a mark for pin1	☐
[assy-size] drawn at the nominal size of the component (*optional*)	☐
[assy-clean] clean of noise; no objects that are not required by 3.7 (*optional*)	☐

Silk layer	ok
[silk-thick] line thicknesses are as recommended: 0.2mm for placement, 0.3mm for pin1 dot (*optional*)	☐
[silk-pin1] pin1 dot is present and is near the right terminal and is outside of component-covered area (*optional*)	☐
[silk-refdes-flg] refdes text is present and is a dyntext floater (*optional*)	☐
[silk-refdes-pos] refdes text is outside of the courtyard (*optional*)	☐
[silk-refdes-hght] refdes text scale is 100% (*optional*)	☐
[silk-style] no excess placement graphics, no full box/frame, no assy drawing on silk	☐
[silk-cy] placement graphics is entirely under the courtyard	☐

Courtyard	ok
[cy-present] there is a courtyard drawing and it contains only polygons	☐
[cy-base1] courtyard covers all component body parts, pins/leads and exposed copper	☐
[cy-base2] "courtyard is not bbox" and does not cover unreasonable bays that are not really used by the component (*optional*)	☐
[cy-extent] courtyard extends beyond base size by the recommended value depending on density (*optional*)	☐
[cy-olap] there are no overlapping or touching courtyard polygons; if there are more than one polygons, they are disjunct and this setup must have a strong, documented reason (*optional*)	☐

Keepout	ok
[ko-present] keepout is present only when there is a strong, documented reason	☐
[ko-olap] there are no overlapping or touching keepout polygons on the same layer; if there are more than one polygons, they are disjunct (*optional*)	☐

Glue (normally omitted)	ok
[glue-style] either line or dots, for center indication	☐
[glue-dim] does not extend beyond component body, with a reasonable margin	☐

3D model (not mandatory)	**ok**
[3d-format] in supported format (OpenSCAD) (*optional*)	☐
[3d-param] if the footprint is generated or parametric, the model is parametric too and the generated footprint contains matching model parameters	☐
[3d-nom] the 3d model represents nominal component dimensions (*optional*)	☐
[3d-lic1] the 3d model comes with the same licenses (use and dist) as the footprint	☐
[3d-lic2] the 3d model use-license does not "contaminate" the exported 3D board model when embedded	☐

Table of Contents