Library Conventions | KiCad EDA Version 3.0.28

Library maintainer rules & guidelines

KLC is suspended until further notice. Rules are currently being revised for KiCad version 6. No version 5 library additions will be accepted.

Revision History

The KiCad Library Convention (KLC) is a set of requirements for contributing to the official KiCad libraries. Users wishing to submit or update library files should be familiar with these guidelines.

The KLC are a set of *guidelines*, rather than *rules*. Electronic component libraries are diverse and complex, and exceptions can be made at the discretion of the library team.

Where the KLC deviates from a particular datasheet or manufacturer recommendation, the datasheet *should take preference* unless there is a good reason not to do so (which should be clarified by a librarian).

Where the KLC is unclear, users should attempt to match the convention of existing library components, or seek further clarification.

Refer to the contribution guidelines for introductory information on contributing to the KiCad libraries.

KLC Helper Scripts

The KiCad library team has developed a set of Python scripts which can be used to help test if library components conform to the KLC requirements.

When a merge request is made to the libraries, the contributed files are automatically checked using these scripts. It can be helpful to run these scripts on your local machine before submitting a PR, as it will help speed up the process of merging your contribution(s) into the library.

To run the footprint checker script, for instance, `cd` into the "kicad-library-utils/pcb" directory, then run the Python script `./check_kicad_mod.py path_to_fp1.kicad_mod path_to_fp2.kicad_mod -vv`. This will carefully check your footprint according to the KLC requirements laid out below, notifying you of any discrepancies, errors, or violations. See more usage examples in the readme at the link above.

Note: While many of the KLC guidelines are checked by these scripts, there are some which are not covered. Additionally, any PR requires manual checking by a member of the library team.

General Library Guidelines

The general library guidelines apply to all library elements (symbols / footprints / models / templates / 3D models). However, these guidelines may be overridden in some cases by specific exceptions described in further sections.

G1 - General Guidelines

G1.1 Only standard characters are used for naming libraries and components

Filenames, symbol names, footprint names, model names, 3D model names, and template names must contain only valid characters, as determined below:

1. Alphanumeric characters (`A-Z`, `a-z`, `0-9`)
2. Underscore `_`

3. Hyphen / dash `-`
4. Period / dot `.`
5. Comma `,`
6. Plus symbol `+`

This character set ensures that symbols will be compatible with all filesystems, and will not cause any issues due to character rendering.

Further, filenames and symbol names must **not use the space character**. This can cause issues with string escaping and is best avoided.

G1.2 Each library is limited to 250 items

Libraries with more than 250 items can result in long library loading times. Additionally, such broad categorization of components means that it can be hard to locate a particular component in the libraries.

A library size limit of `250` items ensures that libraries are quick to load and components are easy to locate. If a given library exceeds `250` components, it should be split and further subcategorized according to component functionality.

G1.3 Libraries are organized by functionality

Rather than grouping components (e.g. symbols, footprints) by their manufacturer, KiCad libraries are organized by component functionality. This grouping strategy has a number of key benefits:

- Similar components are grouped together, allowing alternative parts to be easily substituted
- Symbol *aliases* for pin-identical symbols can be used to reduce library size
- Generic parts which are produced by multiple manufacturers are supported

Library organization should follow the general form as described below, with each element separated by the underscore (_)

character:

1. Library function
2. Library sub-function
3. Tertiary qualifier
4. Manufacturer name
5. Component series name
6. Extra library descriptors

Note: Some of the elements listed above may be omitted if not required.

For specific examples, refer to the guidelines for organizing symbol libraries and footprint libraries.

G1.4 English language should be used throughout libraries

KiCad software is used by people who speak many different languages. The KiCad software provides translations for these languages, but *the library files do not provide translations*.

Library files should be written using English, excepting where specific component names are non-English (for non-English manufacturers).

Where potential differences in spelling and grammar exist, American English should be given preference.

- `Color` instead of `Colour`

G1.5 Plural naming is to be avoided

In general, plural naming (e.g. for libraries) should be avoided, as pluralisation of some words is inconsistent. Non pluralized names should be used in preference:

- `Sensor_Temperature` instead of `Sensors_Temperature`
- `Memory_Flash` instead of `Memories_Flash`
- `TerminalBlock` instead of `TerminalBlocks`

G1.6 Capitalization conventions

1. Acronyms should be capitalized
    - `MCU` (microcontroller)
    - `FPGA` (field programmable gate array)
2. Manufacturer names should be capitalized according to the manufacturer
    - `Microchip`
    - `ROHM`
    - `Texas`
    - `NEC`
3. Unless separated by delimiters such as `_`, `-`, `.`, words should be separated using *CamelCase* convention, with the first letter of each word capitalised
    - `TestPoint`
    - `BatteryHolder`

G1.7 Library files use Unix style line endings

Library files must always be commited to GitLab using Unix-style line endings. This means that lines are terminated using the **LF** (line feed) character, rather than the DOS style **CR+LF** line endings. KiCad library files must be compatible across multiple operating systems, and users should be able to contribute without generating unnecessary *noise* in the files by constantly changing line endings. When using the KiCad libraries on a Windows PC, the line endings in the library files may be automatically converted to **CR+LF**. This is fine, as long as any contributions made to the libraries observe the **LF** line ending requirements.

This issue is addressed by insisting that all KiCad libraries include a .gitattributes file. This file ensures that the line endings are automatically converted by Git.

An example of a simple .gitattributes file which ensures footprint files have correct line endings is as follows:

```
*.kicad_mod text=auto
```

Note: The KiCad library repositories should already have the correct .gitattributes file in place to achieve this

G1.8 Contributions to the official library must be made using the current stable version

Contributions of library assets must be created in the latest **stable** version of KiCad.

This is to avoid problems caused by changes in how assets are stored in different versions of KiCad. (Changes in how line ordering is determined, how strings are escaped, etc.) Another reason is to avoid including features not supported by the latest stable release.

G1.9 Dimensional units

1. When specifying a dimension (e.g. in a footprint name) the *units* must always be provided:
    - `3mm` - millimeters
    - `1in` - inches
2. When specifying multidimensional units (e.g. `length` x `width` x `height`) then the units only need to be appended once
    - `3x4x7mm` - 3mm x 4mm x 7mm
3. Metric units are preferred, where appropriate
4. Linear dimensions should be specified in millimeters
5. Angular dimensions should be specified in degrees
6. Temperature should be specified in degrees Celsius

G2 - Generic and Fully Specified Symbols

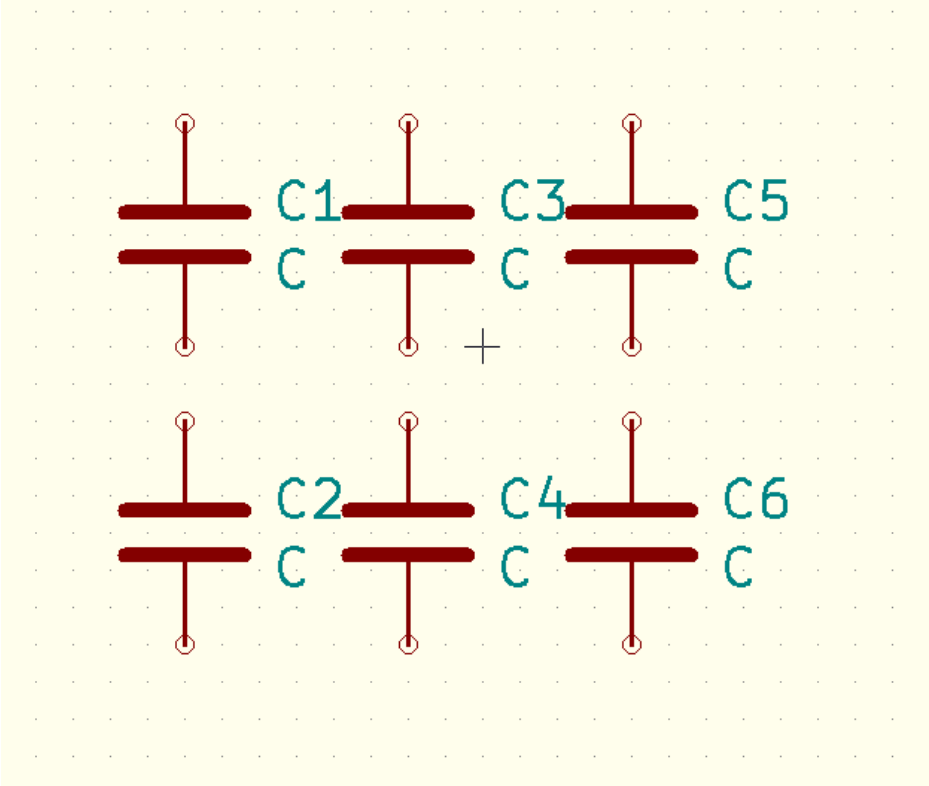
G2.1 Definition of terms: Generic and fully specified symbols

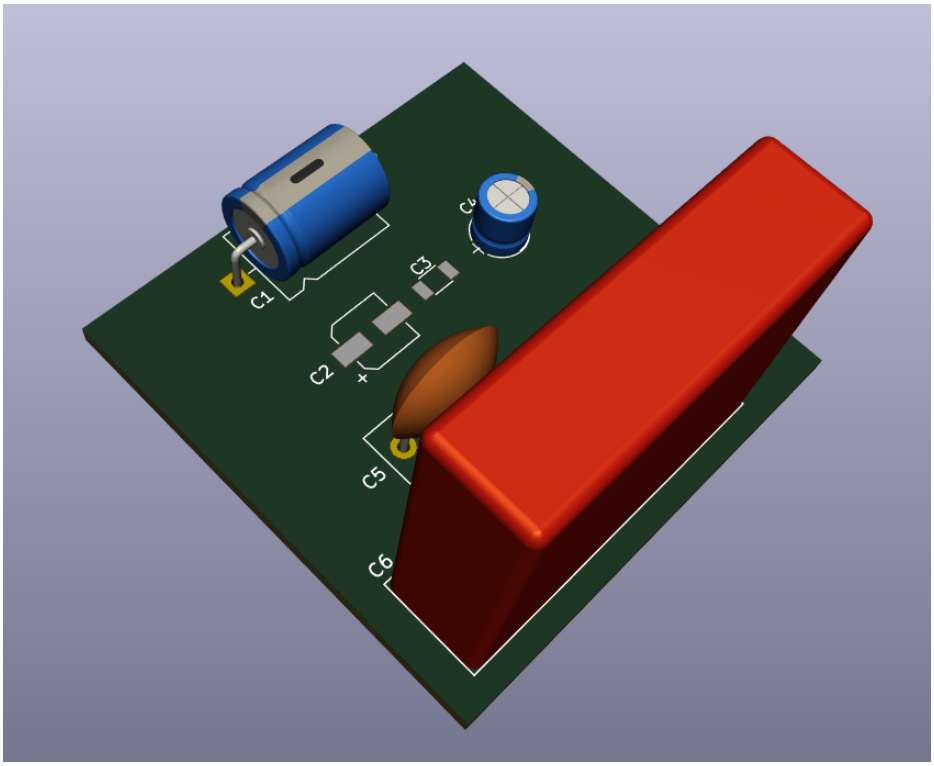
KiCad has two workflows regarding what a symbol represents and when it is assigned a footprint. It can be done in the library in which case the symbol represents an exact part or using the "Assign Footprints" tool at the end of the design process.

Generic symbols

Generic symbols can be used with multiple footprints. They *do not* have a default footprint assigned. Generic symbols allow flexibility in the design workflow. For this workflow the symbol placed into the schematic is typically selected to fit the intended function instead of fitting a particular part. The footprint assignment happens later when the exact part number is selected. Using generic symbols allows a small number of library symbol elements to represent a *very large number* of possible component combinations.

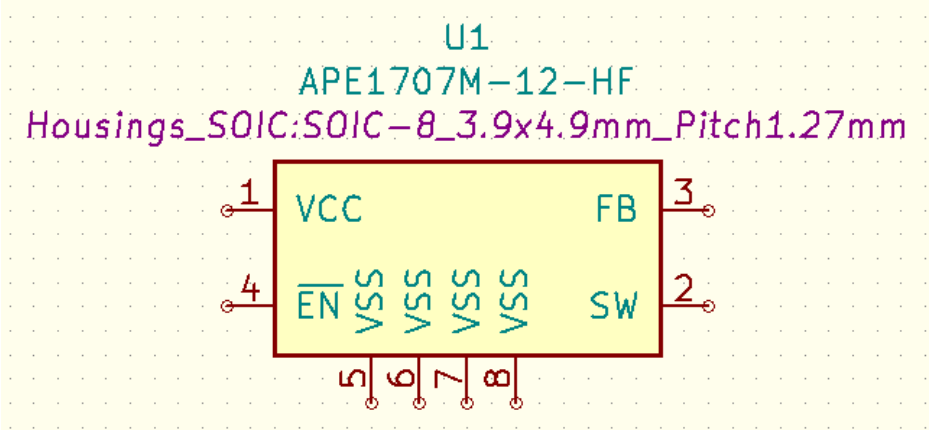
An example of the use of generic symbols are the **Resistor** and **Capacitor** symbols available in the KiCad library. These symbols do not have an assigned default footprint, as there are *many* possible compatible footprints.





Fully specified symbols

Fully specified symbols define a matching footprint, and are named based on the `MPN` (manufacturer part number). Such symbols do not depend on the user to select the correct footprint later but can have alternative footprints defined using footprint filters such that the user can select an alternate footprint.



A fully specified symbol together with a footprint specialized to one component are called an atomic part. A library of only such parts is called an atomic library. Confusingly, some users refer to a fully specified symbol as an atomic symbol.

Where to use which type

The official library requires fully specified symbols except for a small number of libraries that contain only generic symbols. These generic libraries are:

- `Device`
- All `Connector` libraries
- The logic family libraries like `74xx`, `4xxx`, ...

Symbol Guidelines

The following guidelines apply to schematic symbols and symbol library files.

S1 - Symbol Libraries

S1.1 Symbol libraries should be categorized by function

Symbol libraries are individual `.lib` files, which have a corresponding `.dcm` file. Both files are required to fully describe the symbols contained in the library. Apart from the file extension, both files *must have the same name (case sensitive)*.

Symbol library names must be defined based on the priority list below, with each element separated by the underscore (`_`) character:

1. Function (e.g. `Sensor`, `Amplifier`, `MCU`)
2. Sub-function (e.g. `Temperature`, `CurrentSense`)
3. Tertiary qualifier (e.g. `CMOS`)
4. Manufacturer name (e.g. `Atmel`, `Infineon`)
5. Symbol series name (e.g. `PIC24`, `STM32`)
6. Extra library descriptors (e.g. `Deprecated`)

Note: Some of the elements listed above may be omitted if not required.

Example symbol library names:

- `Power_Monitor` - Power monitoring components
- `Sensor_Temperature` - Temperature sensors
- `Driver_Motor` - Motor drivers
- `MCU_Microchip_PIC32` - PIC32 microcontrollers from Microchip

S2 - Symbol Naming

S2.1 General symbol naming guidelines

1. Library naming should not be duplicated in symbol name
2. If symbol with same name exists for multiple manufacturers, the manufacturer name is written first
3. Fully specified symbols are named based on the manufacturer part number
    a. If multiple manufacturers produce compatible parts but their part number suffixes for specifying the package differ, then the common base part number is used with clear name suffix for the footprint (e.g. `L78L05_TO92`)

   b. Non-functional parts of the MPN must be replaced with a wildcard. (Refer to requirements for the use of wildcards in MPN)
4. Generic symbols use the device type
   a. May be shortened for common components (e.g. `Conn` for `Connector`)
   b. Reference designator may be substituted for common components (e.g. `D`, `C`, `LED`)
5. Indicate quantity of elements for symbol arrays (e.g. resistor array with 8 elements - `Resistor_x8`)
6. Any modification of the original symbol, indicated by appending the reason
   a. Different pin ordering - `Q_NPN_CBE`, `Q_NPN_BCE`
   b. For multi-unit variants of typically single-unit symbols add the suffix `_Split`

S2.2 Non-functional variations in part number should be replaced with wildcard

Manufacturer Part Numbers (MPN) often include variations that are *non-functional*.

Examples of parameters considered non-functional:

- Temperature rating of the component
- Packaging information (e.g. reel, tray, tape)
- RoHS / PbFree information

To capture every possible combination of part variation would require a large number of symbol aliases and is to be avoided.

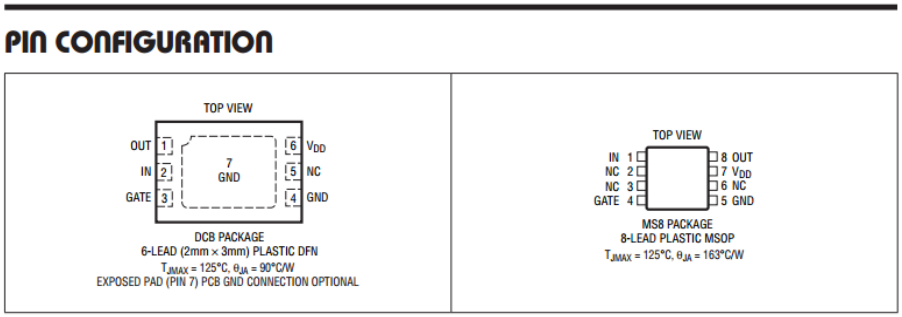
Where an MPN has options for such non-functional variations, these portions of the MPN should be replaced with a wildcard character (`x`). If these non-functional parts of the MPN are at the end of the symbol name then the wildcard should be omitted.

S2.3 Where parts are available in multiple footprint options, a separate symbol must be drawn for each footprint

Many electronic components are provided in multiple packages. These *may or may not* be pin compatible.

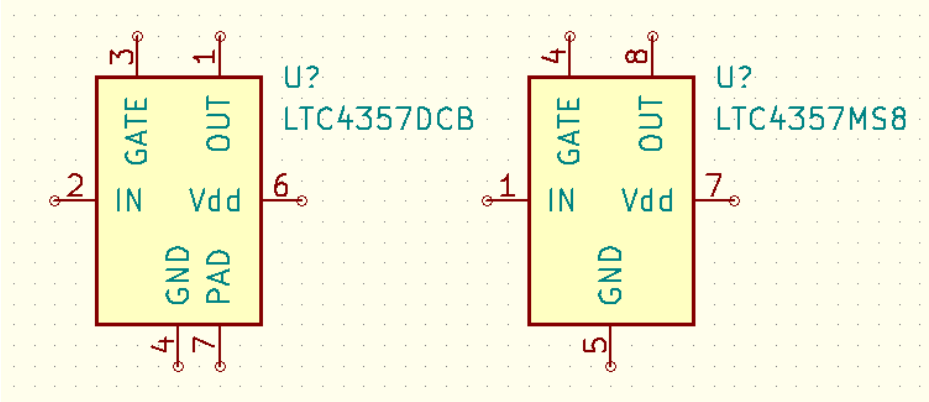
KiCad is not able to assign different footprints for aliases. Fully specified symbols therefore require a separate symbol for every package. Footprints in KiCad have a 1:1 relationship with their 3D model. This means multiple footprints are required where the 3D model has functional differences even if the footprints are identical (for example when 3D model height differs).

e.g. **LTC4357**



This part comes in two distinct packages each requiring a separate footprint. Hence a symbol for each variant must be drawn.

For naming of the symbols refer to link:/symbol/s2/s2.1/ [General symbol naming guidelines]

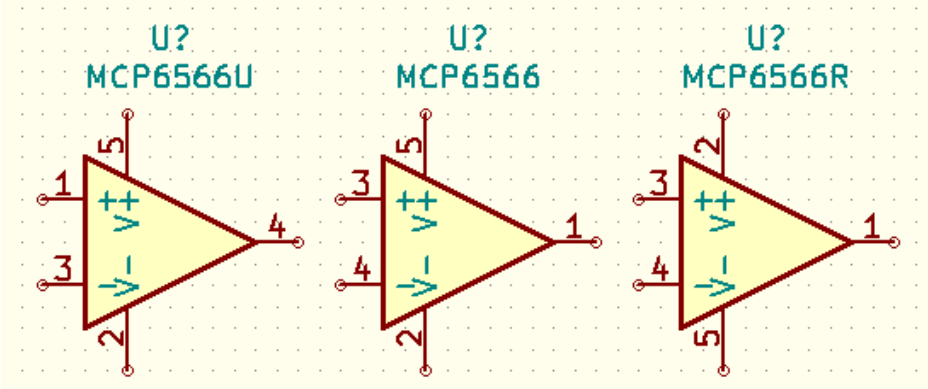


Where possible, the symbols should be drawn such that they can be swapped in the schematic with minimal disruption to wire

connections.

As a further example we shall consider the comparator **MCP6566** which is available in three SOT-23-5 versions, *each with a different pinout*.

In this case, a separate symbol must again be drawn for each version, and named according to the convention called out in the datasheet.



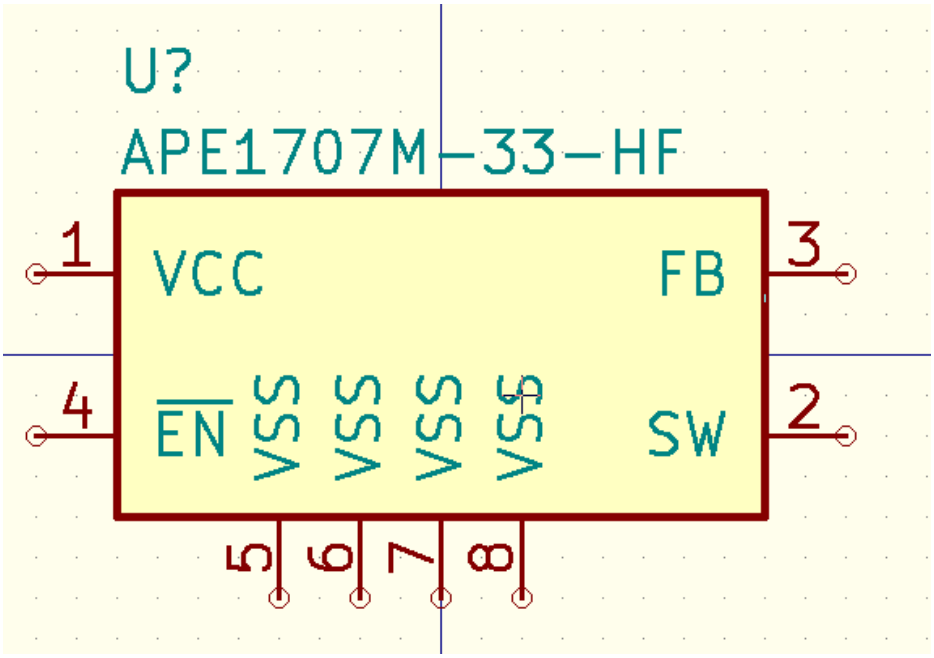
S3 - General Symbol Requirements

S3.1 Origin is centered on the middle of the symbol

For symmetrical symbols, the symbol must be centered around the origin (`0` , `0`) in the symbol editor.

Exception:

For non-symmetrical symbols, or symbols where this requirement would move pins off the `100mil` grid, then the symbol should be placed *as close to the origin as possible* while observing the pin grid alignment requirements.



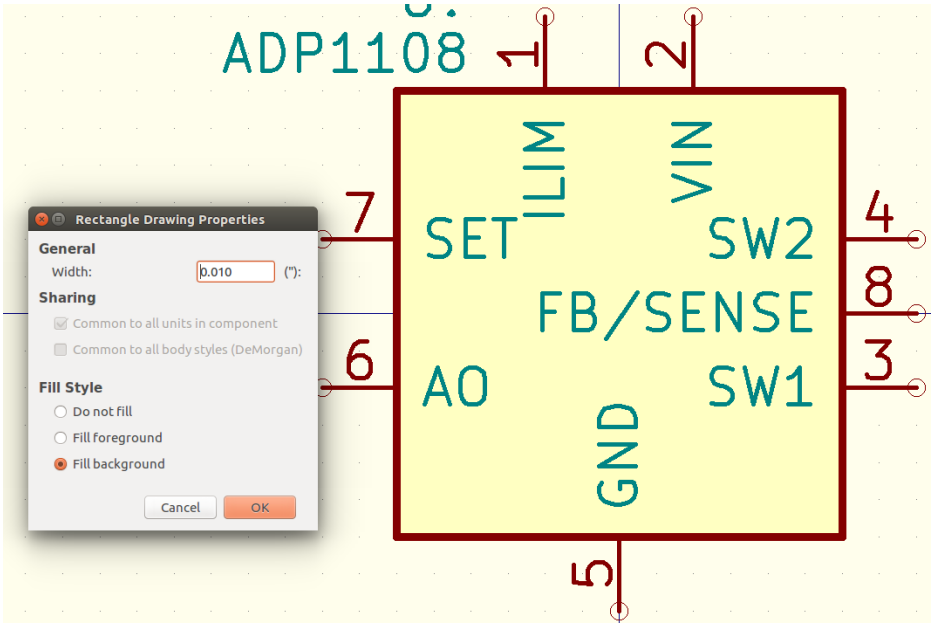
S3.2 Text fields should use a common text size of 50mils

1. All text fields (pin name, pin number, value, reference, footprint, datasheet) should use a text size of `50mil` (`1.27mm`).
2. Pin names and pin numbers can use a text size as small as 20mil if the symbol is small or has special geometry.

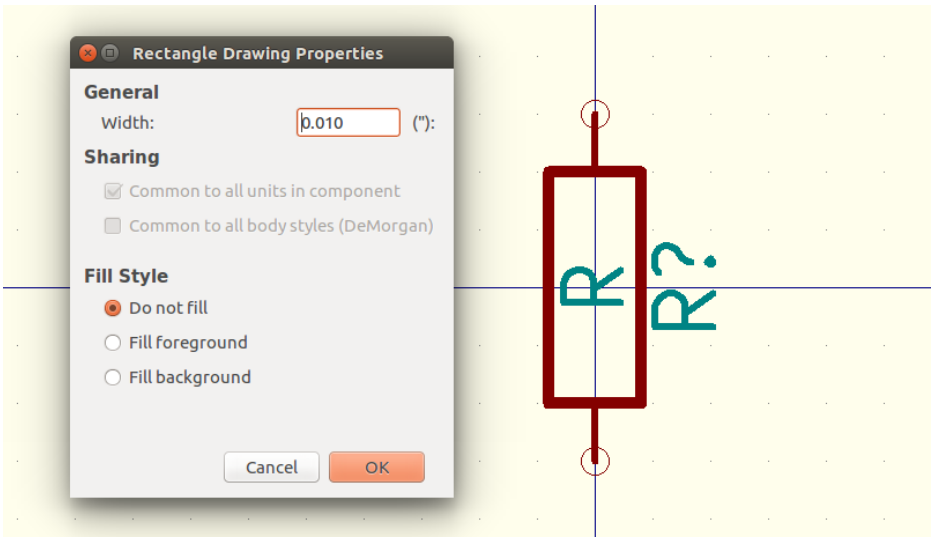
S3.3 Symbol outline and fill requirements

1. Symbol body must have a line width of `10mil` (`0.254mm`)
2. Black-box symbols (e.g. ICs with hidden functionality) should be filled with background color
3. Simple components (e.g. discrete components or those with a *distinctive shape*) should not be filled with background color

Fill background



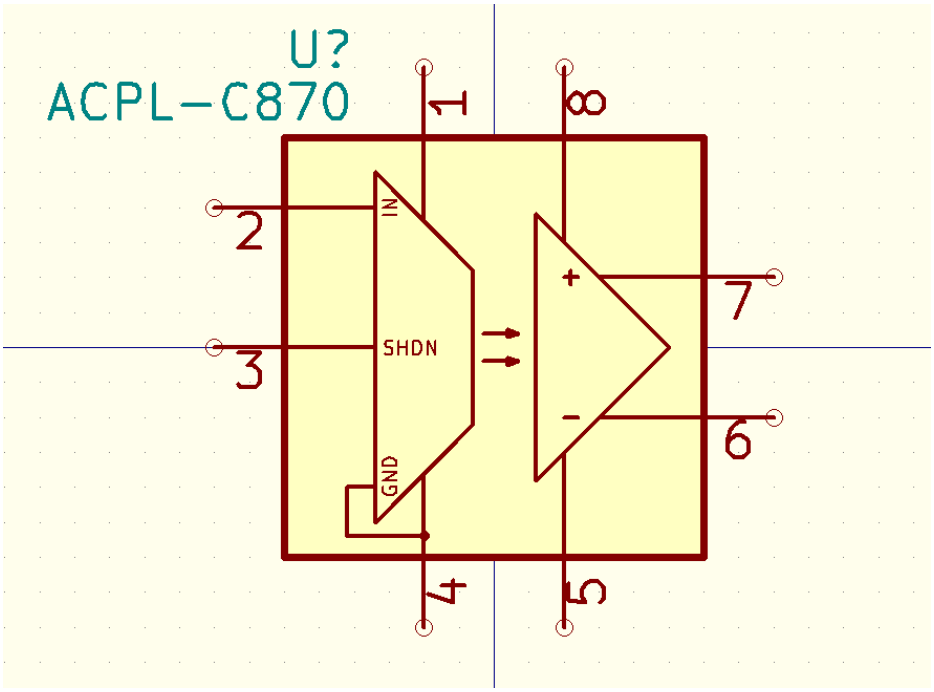
Do not fill



S3.4 Symbols with complex functionality may incorporate simple functional diagrams

For *black box* symbols with complex functionality (e.g. special function ICs), a simplified diagram of the symbol functionality may be drawn on the symbol. This may be used to provide functional clarity in a schematic.

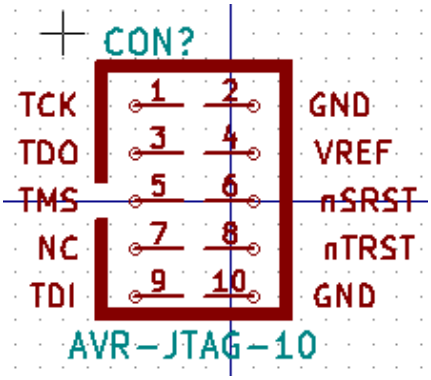
1. Functional drawing must remain *simple*
2. Do not add functional drawings for multi-function components (e.g. microcontrollers)



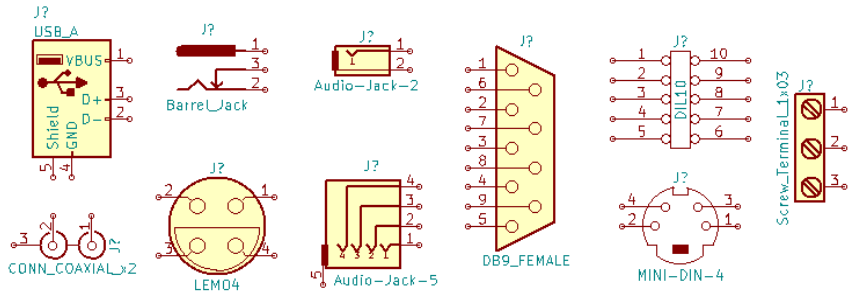
S3.5 Pin connection points must be placed outside of the symbol

Connection points for symbol pins must be placed outside the symbol body and must not require a connecting wire to cross the symbol.

Example of incorrect symbol:



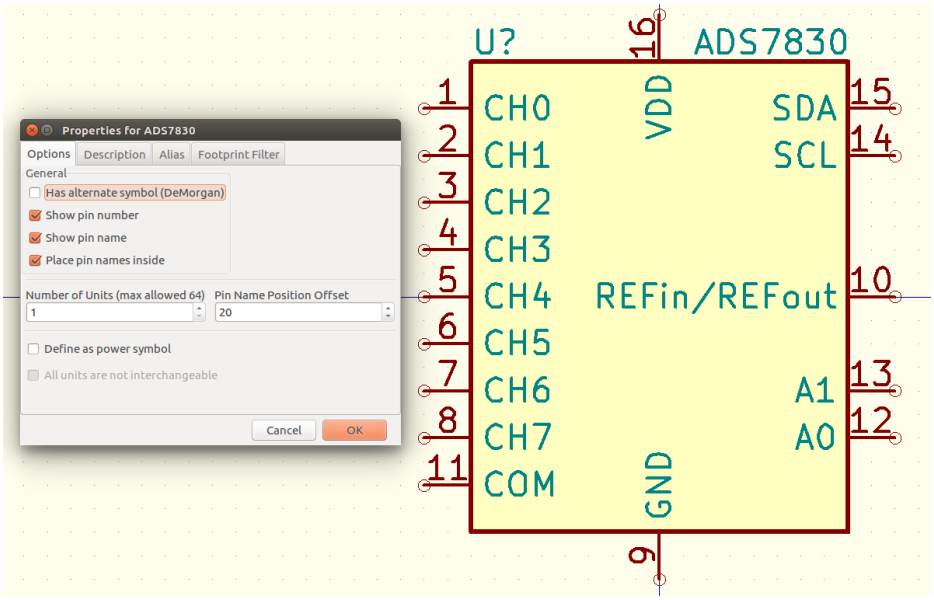
Example of correct symbols:

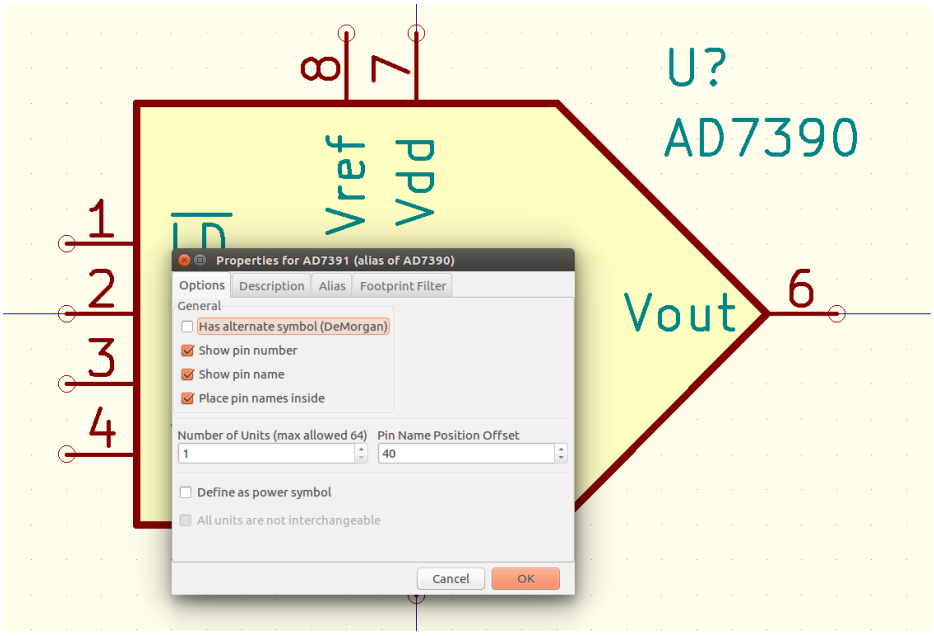


S3.6 Pin name position offset

1. Pin name position offset must not exceed `50mil (1.27mm)`
2. Pin name position offset should not be set less than `20mil (0.508mm)`
3. Preferred value is `20mil`
4. Larger offset allowed to accomodate particular symbol geometry

Examples:





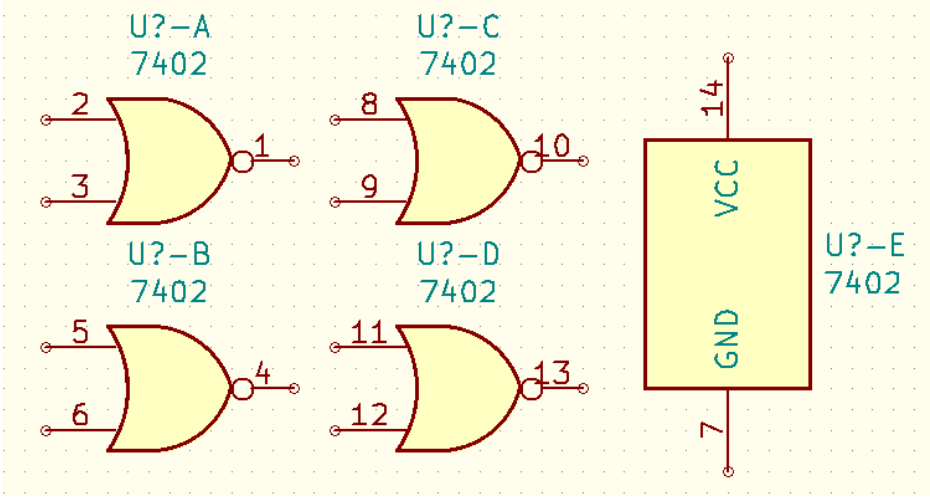
S3.7 Pin numbering for specialized pads

For IC components with exposed pads, the number of the exposed pad should start one greater than the pin-count of the footprint.
For a SOIC-8 package with a single exposed pad, the exposed pad will be assigned the number `9`

Shielded components use "SH" as the pin number for connecting the shield. For a pin to qualify as the shield pin it must be connected to a metal enclosure. For connectors it must connect to the shield of the cable.
Parts that include mechanical mounting pads without electrical use, get the pin number "MP". Non plated through holes do not get any pad number assigned. (They do not get a pin in the symbol.) An example for such a pure mechanical mounting pad would be the pads for the two metal leads on the side of Molex picoblade smd connectors.

S3.8 Multi unit symbols

1. For symbols with multiple units that are drawn separately, where the units share common power pins, a separate unit should be drawn which contains these power pins.



This approach should not be taken for single-unit symbols. Power pins in this case must be drawn on the same symbol as the logic pins.

S4 - Pin Requirements

S4.1 General pin requirements

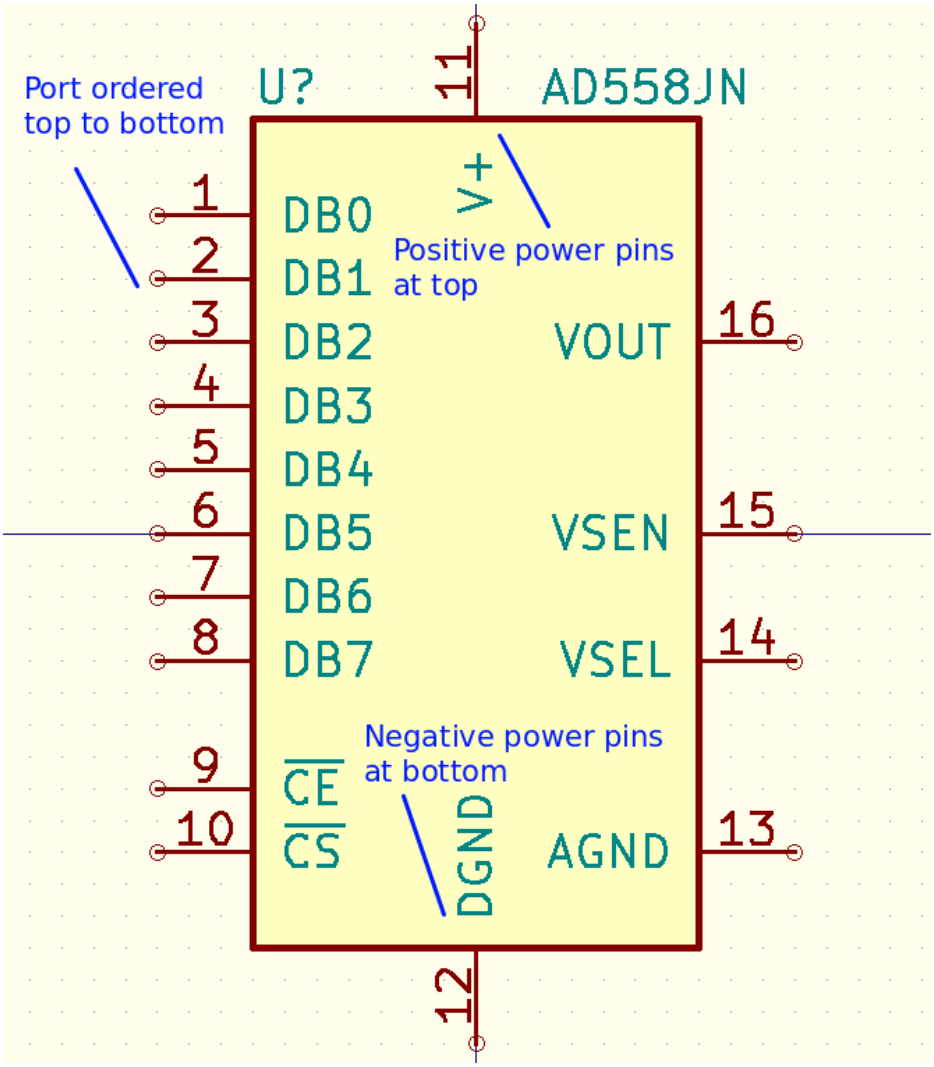
1. Using a `100mil (2.54mm)` grid, pin origin must lie on a grid node (IEC-60617)
2. Pins should have a length of at least `100mil (2.54mm)`
    - Pin length can be increased in steps of `50mil (1.27mm)`
    - The number of characters in the pin number determine a pin's length. When there are two characters in the pin number then `100mil (2.54mm)` long pins should be used, when three characters then `150mil (3.81mm)` long pins should be used, etc.
    - Pin length must not exceed `300mil (7.62mm)`
    - Shorter pins are allowed on simple symbols for discrete devices such as resistors, capacitors, etc.
    - All pins on a symbol must have the same length
3. Pin numbers must be unique (no two pins may have the same number)

S4.2 Pins should be grouped by function

Where possible, pins should be grouped by similar function, rather than by their physical location on the associated footprint. This provides cleaner schematic routing and symbols are easier to understand in the schematic.

1. Pins with **similar functions** should be grouped together:
    - `SPI_MISO`, `SPI_MOSI`, `SPI_CS`, `SPI_CLK`
    - `UART_TX`, `UART_RX`
2. **Ports** should be ordered from top to bottom
3. **Positive power** pins should be placed at the *top* of a symbol
    - `Vcc`, `Vdd`, `Vin`, `V+`, etc.

4. **Negative power** and **ground** pins should be placed at the *bottom* of a symbol
    - `GND`, `Vss`, `V-`, etc.
5. **Input/Control/Logic** pins should be placed on the *left* of a symbol
    - `UART Tx/Rx`
6. **Output/Controlled/Driver** pins should be placed on the *right* of a symbol
    - `RS232 Tx/Rx`



S4.3 Rules for pin stacking

Many symbols have corresponding footprints where multiple physical pins are connected to a single logical net. It is desirable that in such cases the user only has to connect a single pin in the schematic, and

it will automatically route to all the physical pins on the PCB. (This is not only done to reduce clutter in the schematic drawing. The main reason is to move some responsibility for correct connections from the circuit designer over to the library.)

KiCad currently has no native method for designating that a particular symbol pin maps to multiple footprint pins. The following guide serves as a workaround for designing such symbols.

In the schematic view, pins that share the same position are considered to be *connected* by the KiCad routing algorithm. Thus, pins can only be placed in the same location under a very specific set of circumstances:

1. Pins must **not** be of type `No Connect` (these pins should never be connected together)
2. Power supply pins must be stacked unless the datasheet specifies the need for decoupling capacitors on every pin.
3. Pins are logically connected in the symbol
4. Pins must have the same name
5. Pins must have the same electrical type
6. One pin in the stack must be *visible* (all other pins set to *invisible*)

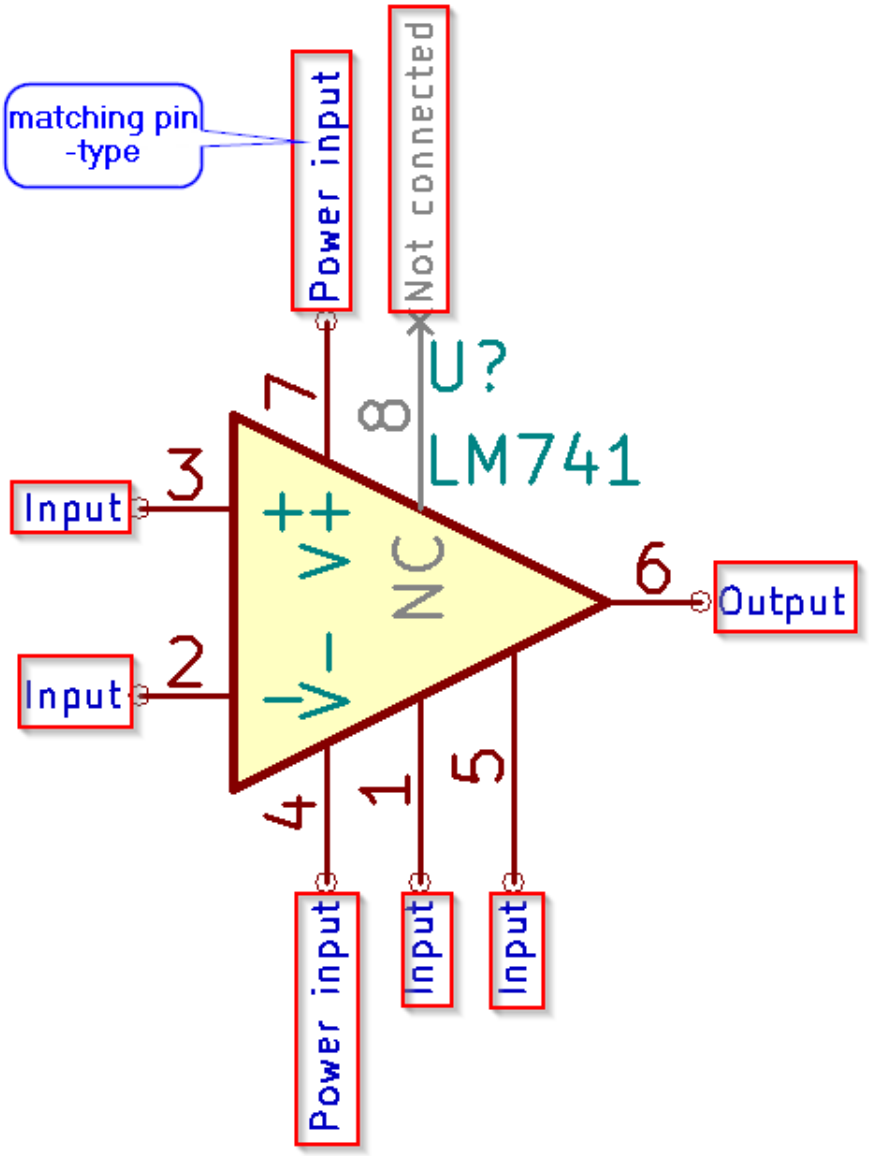
Special Case: Pins of electrical type [`Output`, `Power Output`, `Power Input`] are special cases.

7. Connecting `Output` or `Power Output` pins would result in an ERC error. Output pins that always need to be connected together must therefore be stacked. The invisible pins get the pin type `passive` in this case.
8. Invisible `Power Input` pins are global labels. This is to be avoided. For this reason the electrical type `passive` is to be used for the invisible pins in such stacks.

S4.4 Pin electrical type

Pin `Electrical type` should be set to match the appropriate pin function

1. Power and ground pins should be set to either `Power Input` or `Power Output`
2. Logic pins should be set according to datasheet requirements
3. Pins with programmable functionality (e.g. MCU I/O ports) should be set to `Bidirectional`
4. Pins must not be 'double inverted' by assigning the inverting graphical symbol and also having a bar above the name of the pin



S4.5 Pins not connected on the footprint may be omitted from the symbol

Often a component has pins that are not physically connected. Even these pins should be included in the symbol. Refer to the requirements for hidden pins.

However in some cases unconnected pins may be omitted from the schematic symbol.

1. Pins may only be omitted if they are never to be connected *under any circumstances*
2. Unconnected pins may only be omitted if including them would make the symbol unnecessarily large.
3. If pins are designated `NC` but the datasheet specifies that these must be pulled to ground or otherwise connected to a specific net, then these **must** be shown on the schematic symbol
4. The footprint filter must include the pad count of the footprint if the symbol has omitted pins (see also requirements for footprint filters)
    - `BGA?144`
    - `QFN?20`

S4.6 Hidden pins

Hidden symbol pins are generally not allowed for schematic symbols. Any connection point must be visible, otherwise unexpected net connections can occur.

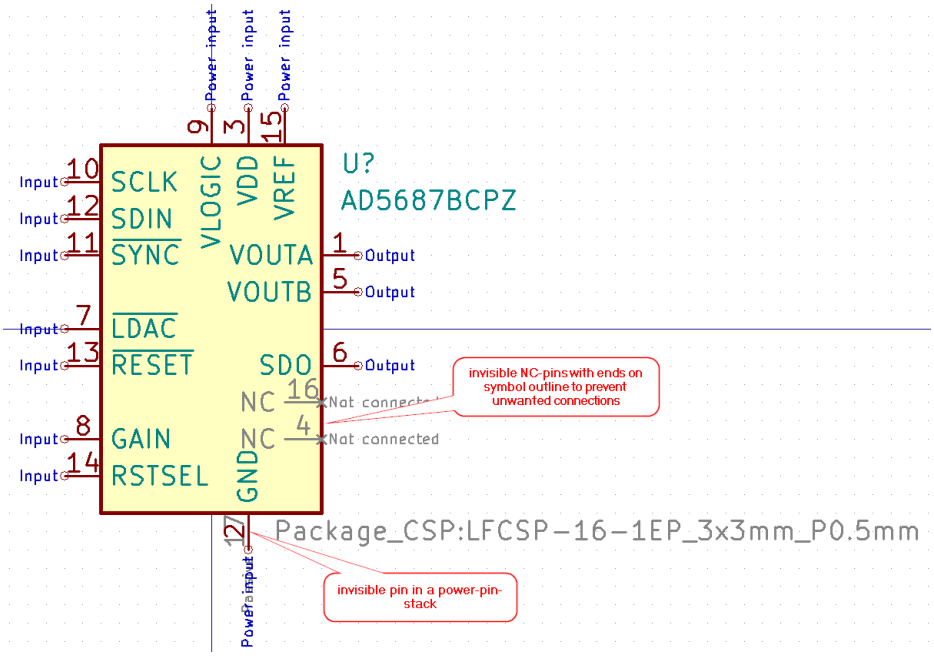
Exceptions

1. Power input pins must not be invisible unless used in power symbols (hidden power input pins are global labels). Refer to the requirements for pin stacking
2. Pins that are not intended to be connected must be set to invisible. In this case the electrical type must be `Not`

`Connected`. The end of the pin should lie within or on the symbols outline to prevent unwanted connections to the invisible pin (see screenshot below).

- Pins specified as not connected may be shown if they are expected to be connected in a significant number of usecases. (Example: The datasheet suggests them to be connected to ground or some other potential for better EMC behaviour.)

3. Invisible pins may be used in a pin stack to allow one-to-many connections. Refer to the requirements for pin stacking



S4.7 Active low pins should be designated using a bar above the symbol name

Rather than selecting the *active low* graphical pin style, active low pins should be designated using a bar above the pin name.

This provides a cleaner look for the schematic.

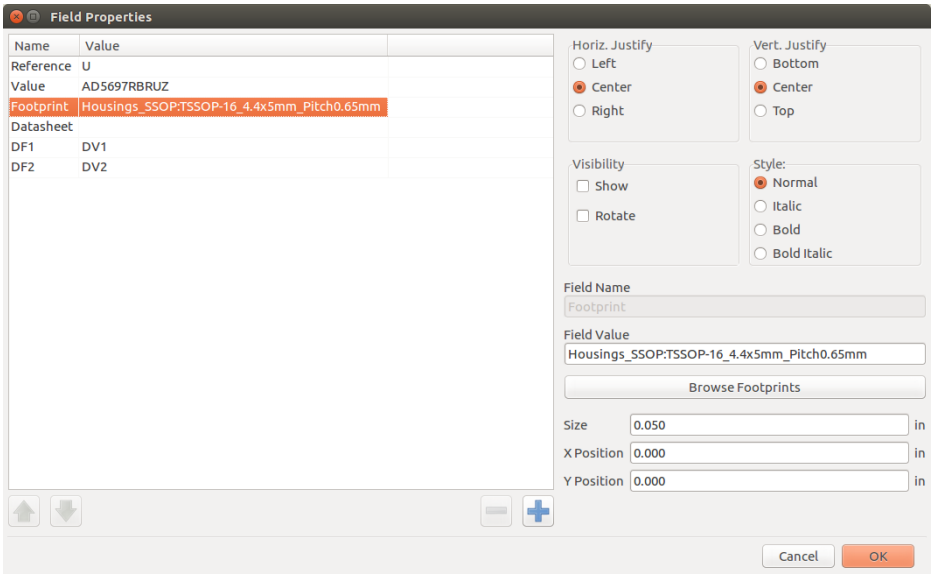
To place a line above the pin text, prefix the name with the tilde (~) character. In case of a pin name where only part of it is active low, and hence, not all of it needs a bar above, two tilde characters must be used as delimitation, e.g. `~SPD_LED~/GPIO2`.

If the pin name given by the manufacturer includes an active low prefix or suffix, it should be removed when adding the bar above it to avoid negating it twice. For example for `nRESET`, use `~RESET`.

S5 - Footprint Association

S5.1 Symbols with a default footprint link to a valid footprint file

1. For *atomic* symbols (those with an associated default footprint), the **Footprint** field must be filled with a valid entry of the format `<footprint_library>:<footprint_name>`.
2. For *generic* symbols (those which map to multiple possible footprints), the **Footprint** field must be left blank



S5.2 Footprint filters must match all appropriate footprints. They should be designed to result in no false suggestions.

Footprint filters are used to help match appropriate footprints to a given symbol. This is important even for *fully specified* symbols (symbols with only one matching footprint), as there still may be

multiple compatible footprints with different variants or options (e.g. `_HandSoldering`, `_Heatsink`, `_ThermalVias`).

Footprint filters use wildcard pattern matching, and allow the following wildcards:

- `*` - Match zero or many characters
- `?` - Match zero or one characters

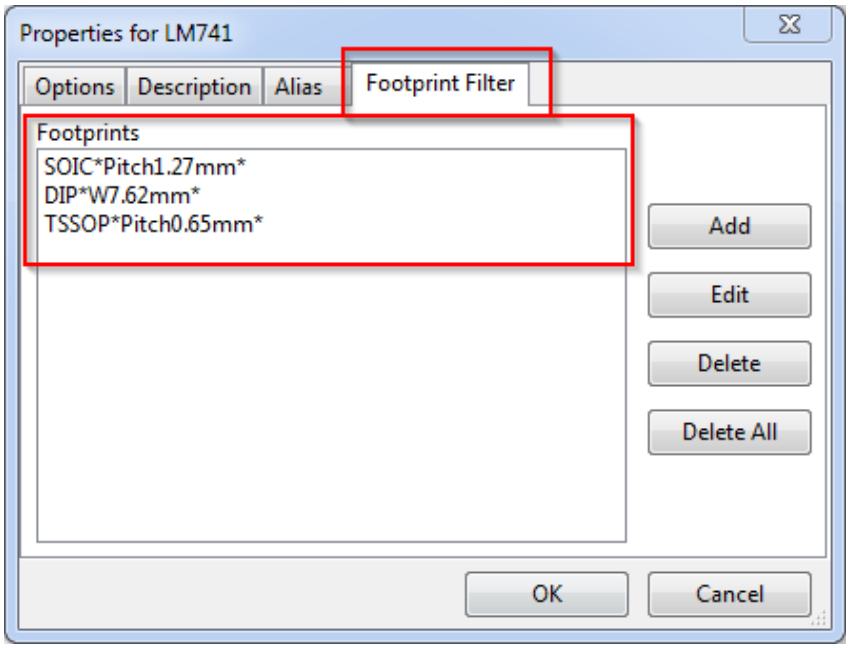
The library name can be used as a filter criterium. The delimiter between library and footprint filtering is `:`

Filter conventions:

1. Filters must end with a `*` wildcard to allow matching of modified footprint suffixes
2. Filters must match the dimensional information (where required) to be as specific as necessary:
    - `DIP*W7.62mm*` to match `DIP-22_W7.62mm` but not `DIP-22_W9.3mm`
    - Neither the size of the exposed pad nor the size of the mask cutout are included.
3. Filters must not contain the pin count if the pin-count in the symbol matches the pin count in the footprint. In those cases the footprint is matched by KiCad's pin count filter. If not all pins are present in the symbol (e.g. NC-pins) the pin-count has to be part of the footprint filter (see requirements for NC pins). Then e.g. `SOT?23?5` might be used instead of simply `SOT?23*` to match `SOT-23-5`.
    - Special pin specifiers are always included. Meaning the specifier for Exposed pads `EP`, Mounting pad `MP` and Shield `SH` including their count must be included if the symbol uses these pins. (Filter for QFN-16-1EP_3x3mm_P0.5mm_EP1.8x1.8mm would be `QFN*1EP*3x3mm*P0.5mm*`)

4. By default, the footprint search does *not* include the name of the footprint library. To force the library name to be included, add the ： (colon) character to the filter. Text appearing before the ： will match the library name. Text appearing after the ： will match the footprint name.
    - `Connector*:*Pitch?1.25mm*` will match any footprint with '1.25mm' pitch, only in libraries that begin with the text 'Connector'

Footprint filters can be set in the **Footprint Filter** tab in the **Symbol Properties** window.



S6 - Symbol Metadata

S6.1 Component Reference Designator (RefDes)

The symbol Reference Designator must be filled out appropriately for the particular *type* of symbol.

The table below provides a list of appropriate RefDes values. If you do not find an appropriate component entry, consult with the KiCad library team.

Designator	Component Type
A	Sub-assembly or plug-in module
BT	Battery
C	Capacitor
D	Diode
DS	Display
F	Fuse
FB	Ferrite bead
FD	Fiducial
FL	Filter
H	Hardware (mounting screws, etc)
J	Jack, fixed part of a connector pair
JP	Jumper / link
K	Relay
L	Inductor, coil, ferrite bead
LS	Loudspeaker or buzzer
M	Motor
MK	Microphone
P	Plug, movable part of a connector pair
Q	Transistor

R	Resistor
RN	Resistor network
RT	Thermistor
RV	Varistor
SW	Switch
T	Transformer
TC	Thermocouple
TP	Test point
U	Integrated circuit (IC)
Y	Crystal / oscillator
Z	Zener diode

S6.2 Symbol and alias fields and metadata filled out as required

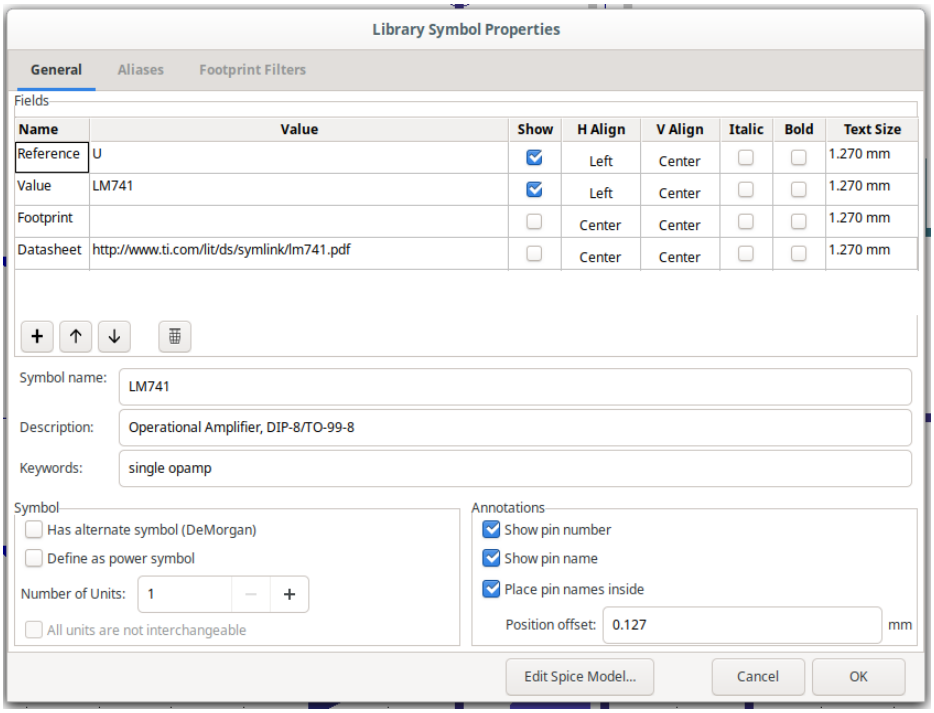
1. **Reference** field is selected appropriately for the symbol and is *visible*
2. **Value** field contains the name of the symbol and is *visible*
3. **Footprint** field contains footprint link for fully specified symbols and must be *invisible*
    a. The footprint field must be empty for generic symbols
4. **Datasheet** entry [1] is filled out, and is *invisible*
5. The symbol contains no other custom fields

Additional documentation is provided via two other fields:

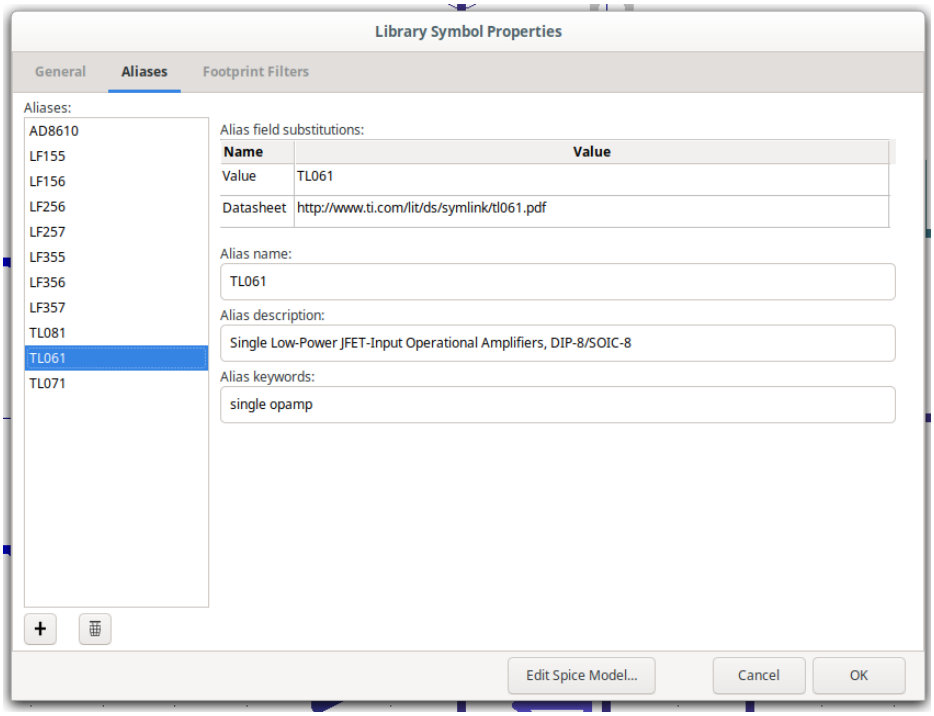
6. **Description** field contains comma-separated device information

   a. For symbols with a default footprint, the simplified footprint name should be appended to the description e.g. `SOIC-8`
   b. Part name should not be duplicated in the description field

7. **Keywords** field contains space-separated keyword values. These values are used to assist in part searching and should not include filler words



Aliases define their own values for the datasheet link, description and keyword entries.



[1]: KiCad 5.x and earlier have two places to store the datasheet link. It can be in the `.lib` file once per symbol or in the `.dcm` file allowing different datasheets per alias. Having different datasheets for aliases only works if the `.lib` entry is left empty. The datasheet entry in the main symbol metadata view is linked to the dcm entry since version 5.1.x.

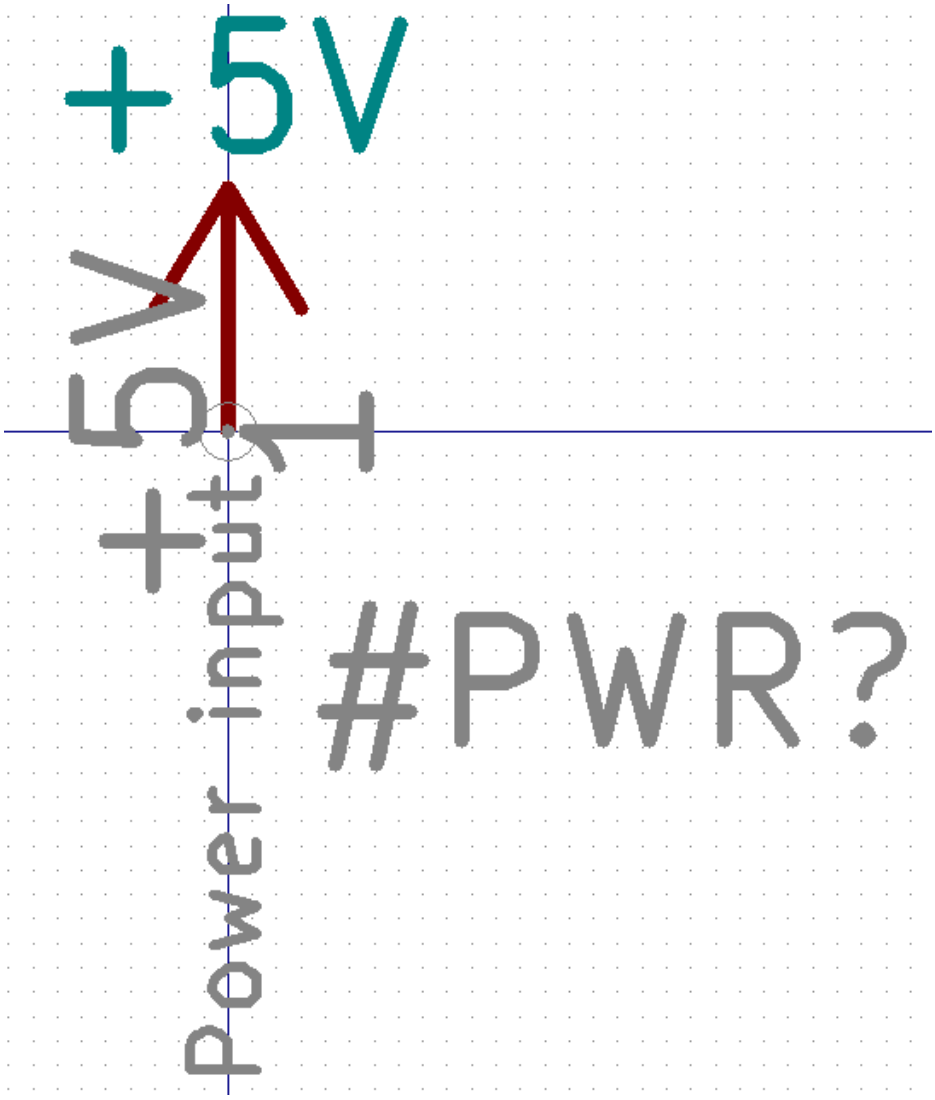
S7 - Special Symbols

S7.1 Power flag symbols

Power flag symbols are special symbols in KiCad, used to designate global nets. These symbols use a special RefDes value to indicate that they must be considered as global power flags

1. Reference Designator must be set to `#PWR`
2. Power flag symbols must contain exactly one pin, which is set to **invisible**
3. Electrical pin type must be set to `Power Input`
4. Pin name must match the symbol name

The example image below demonstrates the requirements as listed above:

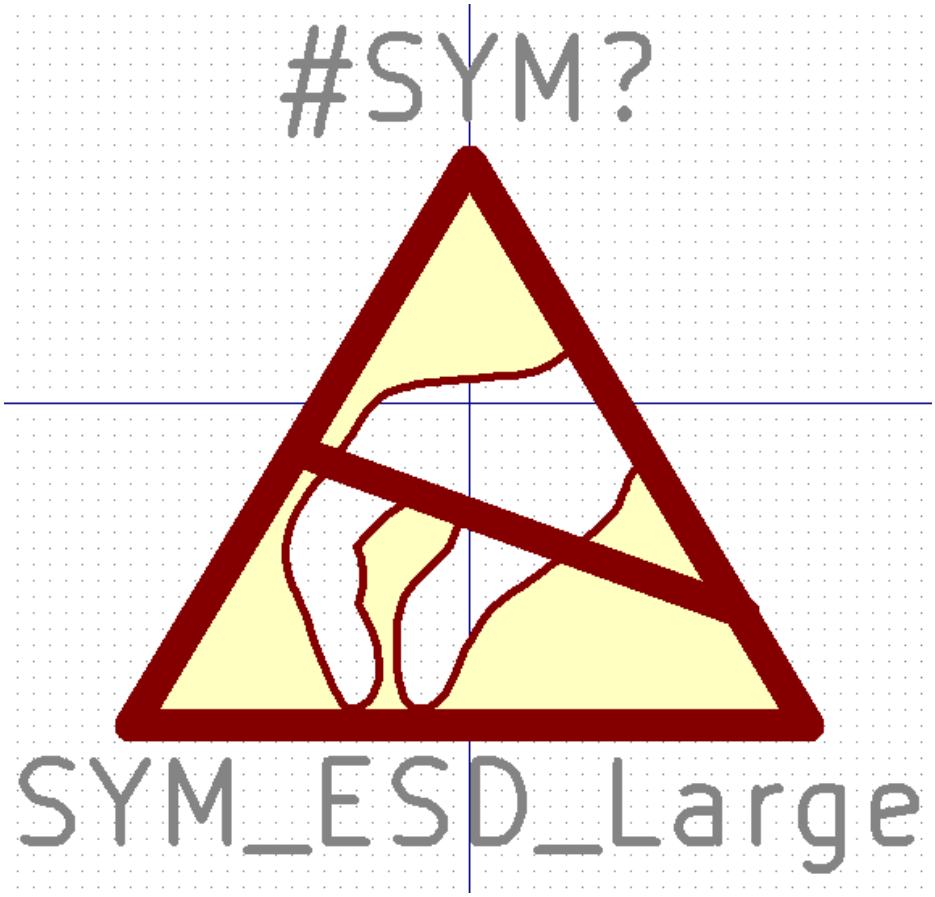


S7.2 Graphical symbols

Graphical symbols are used only to annotate schematics and do **not** correspond to a footprint on the PCB.

1. Reference Designator must be set to `#SYM` and must be **invisible**
2. Symbol name must be set to *invisible*
3. Graphical symbols must not contain any pins
4. No pins or footprint association
5. No footprint filters

Example:



Footprint Guidelines

The following guidelines apply to PCB footprints and footprint libraries.

F1 - Footprint Libraries

F1.1 Footprint libraries are categorized by function

Footprints are grouped into libraries (directories with `.pretty` extension) based on their primary function. As an extension of this, footprint libraries should be named based on this same functionality. Library names must be defined based on the priority list below, with each element separated by the underscore (_) character:

1. Function (e.g. `Conn`, `Capacitor`)
2. Sub-function (e.g. `HDMI`, `USB`)

3. Tertiary qualifier (e.g. `RightAngle`, `SMD`)
4. Manufacturer name (e.g. `Texas`, `Microchip`)
5. Footprint series name (e.g. `MicroFit`)
6. Extra library descriptors as required

Note: Some of the elements listed above may be omitted if not required.

Example footprint library names:

- `Conn_USB_SMD` - Surface mount USB connectors
- `Conn_Molex_MicroFit` - Molex MicroFit series rectangular connectors
- `Capacitor_Tantalum_SMD` - Surface mount Tantalum capacitors

F1.2 Connector footprint libraries

Categorizing connectors *by function* can prove difficult as many connectors can be used in a wide variety of applications. Therefore, categorization of connector footprints is given special consideration as detailed below:

1. Connectors are grouped firstly by their *primary function* (where such a function exists). Examples of *primary function* include:
    - `Conn_USB`
    - `Conn_HDMI`
    - `Conn_Ethernet`
2. When primary function libraries become too large, they can be split by sub-function or by manufacturer
    - `Conn_USB_Molex`
    - `Conn_DSUB_HighDensity`
3. For connectors without a specific function, then connectors are grouped by their *mechanical type*:
    - `Conn_Barrel`
    - `Conn_DIN`

- `Conn_DSUB`
- `Conn_PinHeader`
- `Conn_PinSocket`
- `Conn_Rectangular`

4. Connector libraries can be further split by another qualifier:
    - `Conn_PinHeader_2.54mm`
    - `Conn_PinHeader_1.27mm`
5. Connector libraries can also be split by manufacturer and series, *for multi-purpose connectors*
    - `TerminalBlock_Phoenix_MKDS`
    - `TerminalBlock_Phoenix_PT`
    - `TerminalBlock_Wago`
    - `Conn_JST`
    - `Conn_Hirose_DF13`

F2 - General Footprint Naming Guidelines

F2.1 General footprint naming conventions

Each footprint is a `.kicad_mod` file (stored within a `.pretty` directory). The naming convention for a given footprint depends largely on the *type* of footprint, however a general guide is presented below:

1. Specific package type is written first, e.g.
    - `QFN` - Quad Flat No-Lead package
    - `C` - Capacitor
2. Package name and number of pins are separated by a hyphen
    - `TO-90`
    - `QFN-48`
    - `DIP-20`

3. Packages with special pads add an identifier to the pin count field separated by a hyphen
    - The field includes the count of uniquely numbered pads of this type.
        - For exposed pads (large copper pad below the part) `[count]EP`
        - For shield pads `[count]SH` (Unless such a pin is already expected for the part. An example would be a HDMI connector.)
        - For pads connecting pure mechanical mounting leads `[count]MP`
    - Examples from the library.
        - `DFN-6-1EP_2x2mm_P0.5mm_EP0.61x1.42mm`
        - `Samtec_LSHM-110-xx.x-x-DV-S_2x10-1SH_P0.50mm_Vertical`
        - `Molex_PicoBlade_53261-0271_1x02-1MP_P1.25mm_Horizontal`
4. Unique *fields* (parameters) in the footprint name are separated by `_` character.
5. Package dimensions are specified as `length` x `width` (and optionally `height`)
    - `3.5x3.5x0.2mm`
    - `1x1in`
    - If necessary for clarity, footprint body dimensions may be prefixed with a leading `B`
6. Pin layout
    - `1x10`
    - `2x15`
7. Pitch is specified with a leading `P`:
    - `P1.27mm` - 1.27mm pitch
    - `P5.0mm` - 5.0mm pitch

8. Modifiers to standard footprint values
    - `Drill1.25mm`
    - `Pad2.4x5.2mm`
9. Orientation e.g. `Horizontal`, `Vertical`
10. Any modification to the original footprint, indicated by appending the reason
    - `_HandSoldering`
    - `_ThermalVias`
    - `_CircularHoles`

Not all of the fields defined above are strictly required for a particular footprint. Additional fields may also be added as needed.

F2.2 Footprint naming field prefixes

A footprint name has to convey a lot of information to clearly specify the purpose and parameters of the footprint. Some fields in footprint names are *common* to many footprints and can be shortened using special abbreviations.

Not all footprints will require the use of these abbreviations - they are provided as a method of standardising the manner in which footprint parameters are called out when encountered.

In many cases, the major dimensions (x/y/z) of a footprint may be specified without a prefix, as the body dimensions are assumed to have the greatest priority in the footprint name. Therefore, the "3.2x5.7mm" part of the example name below requires no 'B' prefix, and is not required to be written as "B3.2x5.7mm".

e.g. `SOIC-8_3.2x5.7mm_P1.27mm`

In cases where potential conflicts exist, however, the body dimensions must be explicitly named with the prefix `B`.

Refer to the table below for accepted prefix abbreviations.

Prefix	Field Description	Notes

B	Body dimensions	• Body size (`X` x `Y` x `Z`) • Height parameter `Z` may be omitted • Prefix `B` may be omitted if the body size is the major dimension used to define the footprint
D	Diameter	• Diameter of major axis for cylindrical component • e.g. `D10mm` - 10mm diameter
H	Height	• Height of component measured from PCB • Body size `B` should be used in preference, where possible • e.g. `H1.1mm` - component measures 1.1mm above PCB
L	Length	• Length along major axis of component • Body size `B` should be used in preference, where possible • e.g. `L15.3mm` - 15mm length
W	Width	• Width along minor axis of component • Body size `B` should be used in preference, where possible • e.g. `W4.75mm` - 4.75mm width
EP	Exposed pad dimensions	• Packages that include a single exposed pad include the size of

		this pad. e.g. `EP2.4x3.6mm` - for a single 2.4mm x 3.6mm pad.
P	Pad pitch	• Pitch (distance) between pins, pads or leads • e.g. `P0.63mm` - 0.63mm pitch between pads
T	Thickness	Component thickness, where appropriate

Examples:

- `LQFP-32_4x4x1.1mm_P1.65mm` - LQFP package, 32 pins, 4x4mm body, 1.1mm height, 1.65mm pitch
- `CP_Radial_D4.5mm_P2.5mm_H10mm` - Radial polarized capacitor, 4.5mm diameter, 2.5mm pitch, 10mm height

If the parameter is not found in the table above, the name of the parameter should be entered in full.

Following is a list of examples of (non-abbreviated) parameter names used in footprint naming

Prefix	**Field Description**
Ball	BGA Ball diameter
Clearance	Clearance between pin rows
Drill	Drill diameter
Layout	Specific footprint layout e.g. `Layout3x7` for a BGA with 21 balls arranged in 3 columns and 7 rows
Lead	Lead dimensions
Mask	Custom soldermask dimensions
Pad	Custom pad dimensions

Tab	Tab dimensions

F2.3 Manufacturer specific version of generic footprints

If a manufacturer footprint deviates from the "standard" footprints available, then this should be added to the library and designated as manufacturer-specific.

1. If a footprint is specific to a single manufacturer, this must be indicated by prepending the manufacturer name and MPN to the footprint name.
    - `Texas_S-PVQFN-N48_`
2. Sometimes the component datasheet will have specific requirements which do not conform to KLC requirements. A common example is specific requirements for the shape of stencil (solder paste layer).

In such cases, the KLC requirements can be superseded by the datasheet requirements, at the discretion of the KiCad library team.

F2.4 Footprint naming for non-standard pin numbering

Generally, a footprint should include a field of the format `<pkg>-<pincount>`, e.g.

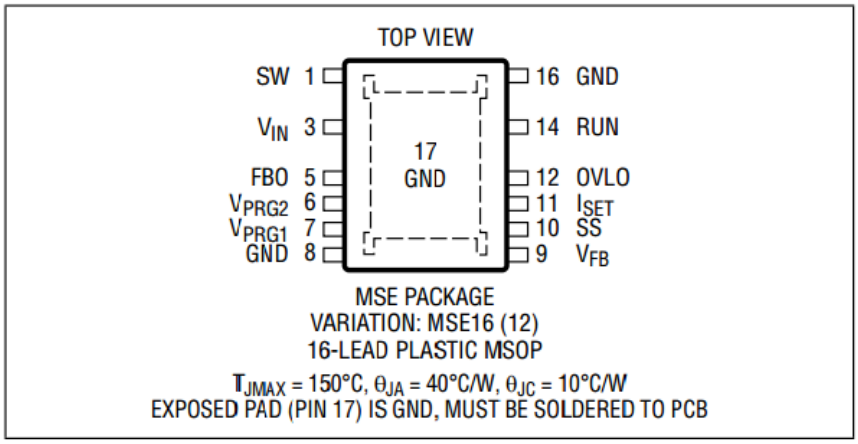
- `SOIC-8`
- `QFN-24`

In the majority of cases, the pin quantity is self explanatory and is sufficient to describe the number of pins on the symbol.
However there may be cases where this numbering is insufficient. Where the number of footprint pins does not match the number of package pins, exceptions must be made.

Exception: Omitted pins, missing pin numbers are skipped

Example: LTC3638

The LTC3638 comes in an `MSOP-16` package which has four missing pins, for increased voltage isolation.



In this case, the numbers for the missing pins are **skipped** and the normal pin numbers are assigned to the remaining pins.

To indicate this, the footprint should be named as follows:

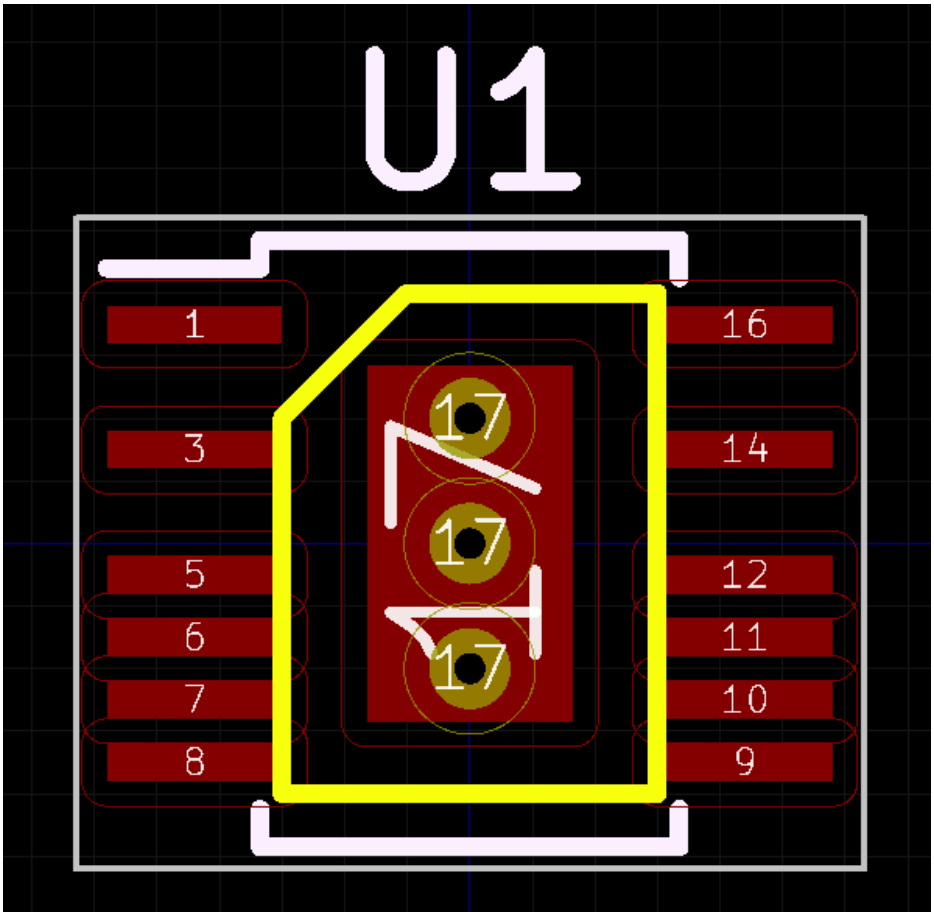
`PKG-<xx>-<yy>` where:

- `xx` = number of remaining pins
- `yy` = number of remaining pins + number of removed pins

In the LTC3638 example above, this would result in:

`MSOP-12-16`

The footprint pads should be numbered like so:



F2.5 Footprint naming conventions for specific components

Footprint naming is sufficiently complex that a general rule is not enough to fully define a naming scheme that fits the wide range of footprints.

In addition to the general footprint naming guidelines defined in KLC 2.1, a set of footprint naming guidelines for specific footprint types is provided in KLC F3.x.

F3 - Specific Footprint Naming Guidelines

F3.1 SMD chip package naming conventions

The following footprint naming conventions should be used as *examples* for naming SMD chip package footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields
- Mandatory fields
- Optional fields

[*Prefix*]_[*Imperial size*]_[*Metric size*]**Metric**_[*Modifiers*]_[*Options*]

- **Prefix** - Device type
- **Imperial size** - Imperial case size (Optional)
- **Metric size** - Metric case size (Optional)
- **Modifiers** - Modifiers to footprint if non standard (Optional)
- **Options** - Extra footprint options (Optional)

Example:

LED_0805_2012Metric_ReverseMount

Notes:

The prefix is determined by the default reference designator of the device, e.g.

- Capacitor = `C`
- Capacitor (polarised) = `CP`
- Resistor = `R`
- Diode = `D`

Case size codes:

Imperial Code	Metric Code	Size Imperial	Size Metric
01005	0402	0.0157 in × 0.0079 in	0.4 mm × 0.2 mm

Imperial Code	Metric Code	Size Imperial	Size Metric
0201	0603	0.024 in × 0.012 in	0.6 mm × 0.3 mm
0402	1005	0.039 in × 0.020 in	1.0 mm × 0.5 mm
0603	1608	0.063 in × 0.031 in	1.6 mm × 0.8 mm
0805	2012	0.079 in × 0.049 in	2.0 mm × 1.25 mm
1008	2520	0.098 in × 0.079 in	2.5 mm × 2.0 mm
1206	3216	0.126 in × 0.063 in	3.2 mm × 1.6 mm
1210	3225	0.126 in × 0.098 in	3.2 mm × 2.5 mm
1806	4516	0.177 in × 0.063 in	4.5 mm × 1.6 mm
1812	4532	0.18 in × 0.13 in	4.5 mm × 3.2 mm
1825	4564	0.18 in × 0.25 in	4.5 mm × 6.4 mm
2010	5025	0.197 in × 0.098 in	5.0 mm × 2.5 mm

Imperial Code	Metric Code	Size Imperial	Size Metric
2512	6332	0.25 in × 0.13 in	6.3 mm × 3.2 mm
2920	7451	0.29 in × 0.20 in	7.4 mm × 5.1 mm

F3.2 Resistor naming conventions

The following footprint naming conventions should be used as *examples* for naming resistor footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields
- Mandatory fields
- Optional fields

Axial Resistors

R_Axial_**L**[*Length*]_**D**[*Diameter*]_**P**[*Pitch*]_[*Modifiers*]_[*Orientation*]_[*Options*

- **Length** - Resistor body length
- **Diameter** - Body diameter
- **Pitch** - Lead spacing
- **Modifiers** - Modifiers to footprint specifications (Optional)
- **Orientation** - Resistor orientation (Vertical / Horizontal) (Optional)
- **Options** - Extra footprint options (Optional)

Example:

R_Axial_L3.6mm_D1.6mm_P2.54mm_Vertical

F3.3 Capacitor naming conventions

The following footprint naming conventions should be used as *examples* for naming capacitor footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields
- Mandatory fields
- Optional fields

SMD Chip Capacitor

[*C/CP*]_[*Imperial size*]_[*Metric size*]**Metric**_[*Modifiers*]_[*Options*]

- **C/CP** - Capacitor style (polarised or non-polarised)
- **Imperial size** - Imperial case size (Optional)
- **Metric size** - Metric case size (Optional)
- **Modifiers** - Footprint modifiers if non standard (Optional)
- **Options** - Footprint options (Optional)

Example:

C_0805_2012Metric

Notes:

1. At least one case size (either imperial or metric) must be specified
2. Metric case size must have the suffix `Metric`

SMD Tantalum Capacitor

CP_Tantalum_EIA-[*Metric size*]_[*Size code*]_[*Modifiers*]_[*Options*]

- **Metric size** - EIA metric size code
- **Size code** - Letter size code including naming standard ([Standard]-[Letter])
- **Modifiers** - Footprint modifiers if non standard (Optional)
- **Options** - Footprint options (Optional)

Example:

CP_Tantalum_EIA-3126-18_Kemet-A_Pad1.53x1.40mm_HandSolder

EIA Metric size	Body size (XxYxZ)	Kemet Code	AVX Code
1608-08	1,6 x 0,8 x 0,8 mm	—	J
1608-10	1,6 x 0,85 x 1,05 mm	—	L
2012-12	2,05 x 1,35 x 1,2 mm	R	R
2012-15	2,05 x 1,35 x 1,5 mm	—	P
3216-10	3,2 x 1,6 x 1,0 mm	I	K
3216-12	3,2 x 1,6 x 1,2 mm	S	S
3216-18	3,2 x 1,6 x 1,8 mm	A	A

EIA Metric size	Body size (XxYxZ)	Kemet Code	AVX Code
3528-12	3,5 x 2,8 x 1,2 mm	T	T
3528-15	3,5 x 2,8 x 1,5 mm	—	H
3528-21	3,5 x 2,8 x 2,1 mm	B	B
6032-15	6,0 x 3,2 x 1,5 mm	U	W
6032-20	6,0 x 3,2 x 2,0 mm	—	F
6032-28	6,0 x 3,2 x 2,8 mm	C	C
7343-15	7,3 x 4,3 x 1,5 mm	W	X
7343-20	7,3 x 4,3 x 2,0 mm	V	Y
7343-30	7,3 x 4,3 x 3,0 mm	—	N
7343-31	7,3 x 4,3 x 3,1 mm	D	D
7343-40	7,3 x 4,3 x 4,0 mm	Y	—

EIA Metric size	Body size (XxYxZ)	Kemet Code	AVX Code
7343-43	7,3 x 4,3 x 4,3 mm	X	E
7360-38	7,3 x 6,0 x 3,8 mm	E	—
7361-38	7,3 x 6,1 x 3,8 mm	—	V
7361-438	7,3 x 6,1 x 4,3 mm	—	U

SMD Electrolytic Capacitor

CP_Elec_[*Manufacturer*]_[*X*]x[*true*]_**H**[*Height*]_**P**[*Pitch*]_[*Modifiers*]_[*Options*

- **Manufacturer** - Manufacturer name (Optional)
- **X** - Case size in x dimension
- **true** - Case size in y dimension
- **[Height]** - Case height (Optional)
- **[Pitch]** - Lead spacing (Optional)
- **[Modifiers]** - Footprint modifiers if non-standard (Optional)
- **[Options]** - Footprint options (Optional)

Example:

CP_Elec_10x10.5mm_H6mm

THT Capacitor

[*C/CP*]_[*Style*]_**L**[*Case size*]_**D**[*Diameter*]_**W**[*Width*]_**P**[*Pitch*]_[*Modifiers*]_[*Options*]

- **C/CP** - Capacitor polarisation
- **Style** - Capacitor case style (see notes below)

- **Case size** - Body length (Optional)
- **Diameter** - Case diameter (Optional)
- **Width** - Body width (Optional)
- **Pitch** - Lead spacing
- **Modifiers** - Footprint modifiers if non-standard
- **Options** - Footprint options (Optional)

Example:

C_Axial_L67.0mm_D26.0mm_P75.0mm

Notes:

1. There are different capacitor styles available:
    - Axial: Cylindrical body with axial lead attachment.
    - Radial: Cylindrical body, both leads enter at same end.
    - Disc: Disc shaped body (sometimes enclosed), leads enter tangent to the inner disc
    - Rect: Box shaped body (Bottom face in contact with the board)
2. Body size:
    - For Axial body styles: L[length]_D[diameter]
    - For Radial body style: D[diameter]{_H[height]}
    - For Disc body style: D[diameter]_W[width]{_H[overall height]}
    - For box shaped body styles: [X]x[Y]{x[Z]}

F3.4 SMD IC package naming conventions

The following footprint naming conventions should be used as *examples* for naming SMD IC package footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields

- Mandatory fields

- Optional fields

Gullwing Packages

The format below provides a guide for naming Gullwing IC packages.

Examples:

- SOIC
- SOP
- QFP
- J-Lead

[*PKG*]-[*Pin count*]-[*EP Qty*]**EP**_[*X*]x[*Y*]x[*Z*]_**P**[*Pitch*]_[*Modifiers*]_[*Options*]

- **PKG** - Package name
- **Pin count** - Number of (unique) pins
- **EP Qty** - Number of exposed pads under the device (if greater than zero) (Optional)
- **X** - Major dimension of part on x axis
- **Y** - Major dimension of part on y axis
- **Z** - Major dimension of part on z axis (Optional)
- **Pitch** - Lead pitch
- **Modifiers** - Footprint modifiers (Optional)
- **Options** - Extra footprint options (Optional)

Example:

SOIC-16[-][1][EP][_][3.9][x][9.9mm][_][P][1.27mm]

Notes:

1. [`PKG`] - refers to the name most commonly used in the industry. Generally this implies JEDEC naming but some other standard may be used as required.

2. [`Pin count`] - the number of *uniquely numbered* pads (excluding any exposed pads under the device)
3. [`Modifiers`] - may vary for individual IC footprint types.

   Includes:

   - Clearance
   - Lead size
   - Pad size
   - Exposed pad size
   - Soldermask expansion

No-Lead Packages

Examples:

- DFN
- QFN
- LCC

[*PKG*]-[*Pincount*]-[*EP Qty*]**EP**_[*X*]x[*Y*]x[*Z*]_**P**[*Pitch*]_[*Modifiers*]_[*Options*]

- **PKG** - Package name
- **Pincount** - Number of (unique) pins
- **EP Qty** - Number of exposed pads under the device (if greater than zero) (Optional)
- **X** - Major dimension of part on x axis
- **Y** - Major dimension of part on y axis
- **Z** - Major dimension of part on z axis (Optional)
- **Pitch** - Lead pitch
- **Modifiers** - Footprint modifiers (Optional)
- **Options** - Extra footprint options (Optional)

Example:

DFN-12[-][1][EP][_][4][x][4][x][0.9mm][_][P][0.5mm]

Notes:

1. The lead length is the nominal lead length for which the footprint is designed
2. Optionally the nominal lead width can also be provided

Ball Grid Array Packages

[*PKG*]-[*Pincount*]_[*X*]x[*Y*]x[*Z*]_**Layout**[*Columns*]x[*Rows*]_**P**[*Pitch X*]x[*Pitch Y*]_**Ball**[*Ball*]_**Pad**[*Pad*]_[*NSMD/SMD*]_[*Modifiers*]_[*Options*]

- **PKG** - Package name
- **Pincount** - Number of (unique) pins
- **X** - Major dimension of part on x axis
- **Y** - Major dimension of part on y axis
- **Z** - Major dimension of part on z axis (Optional)
- **Columns** - Number of ball columns
- **Rows** - Number of ball rows
- **Pitch X** - Ball pitch in x direction
- **Pitch Y** - Ball pitch in y direction (if different from x direction (Optional)
- **Ball** - Ball diameter
- **Pad** - Pad diameter
- **NSMD/SMD** - Solder Mask Defined or Non Solder Mask Defined
- **Modifiers** - Footprint modifiers (Optional)
- **Options** - Extra footprint options (Optional)

Example:

BGA-24[_][3.2][x][4.4mm][_][Layout][3][x][4][_][P][0.5mm][_][Ball][0.25mm][_][Pad][0.2mm][_][NSMD]

Notes:

1. For BGA simply providing the pin count is not sufficient. It is also necessary to provide the number of [`Columns`] (x direction) and [`Rows`] (y direction).

2. Some BGA packages have different pitch in x and y directions. If this is the case then both [`Pitch`] dimensions need to be provided
3. The [`Ball`] diameter for which the footprint is designed needs to be provided
4. [`NSMD/SMD`]
    - NSMD means the footprint is designed for `non solder mask defined` (mask cutout is larger than copper pad)
    - SMD means the footprint is designed for `solder mask defined` (mask cutout is smaller than copper pad)

F3.5 THT IC package naming conventions

The following footprint naming conventions should be used as *examples* for naming THT IC package footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields
- Mandatory fields
- Optional fields

Through Hole Packages (DIP)

[*PKG*]-[*Pincount*]_[*X*]x[*Y*]x[*Z*]_**W**[*Lead span*]_**P**[*Pitch*]_[*Modifiers*]_[*Options*]

- **PKG** - Package name
- **Pincount** - Number of (unique) pins
- **X** - Major dimension of part on x axis
- **Y** - Major dimension of part on y axis
- **Z** - Major dimension of part on z axis (Optional)

- **Lead span** - Distance between leads
- **Pitch** - Lead pitch
- **Modifiers** - Modifiers to footprint specifications
- **Options** - Extra footprint options (Optional)

Example:

DIP-14[_][W][10.16mm][_][Clearance]

Notes:

1. Drill size (if specified) must be given in `mm`

Through Hole Packages (TO)

[*PKG*]-[*Pincount*]_[*X*]x[*Y*]x[*Z*]_**W**[*Lead span*]_**P**[*Pitch x*]x[*Pitch y*]_[*Stagger*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **PKG** - Package name
- **Pincount** - Number of (unique) pins
- **X** - Major dimension of part on x axis (Optional)
- **Y** - Major dimension of part on y axis (Optional)
- **Z** - Major dimension of part on z axis (Optional)
- **Lead span** - Distance between leads (Optional)
- **Pitch x** - Lead pitch in x dimension (Optional)
- **Pitch y** - Lead pitch in y dimension (if different from x dimension) (Optional)
- **Stagger** - Pin staggering options (Optional)
- **Modifiers** - Modifiers to footprint specifications (Optional)
- **Orientation** - Component orientation (see notes below)
- **Options** - Extra footprint options (Optional)

Example:

TO-220-5_Vertical

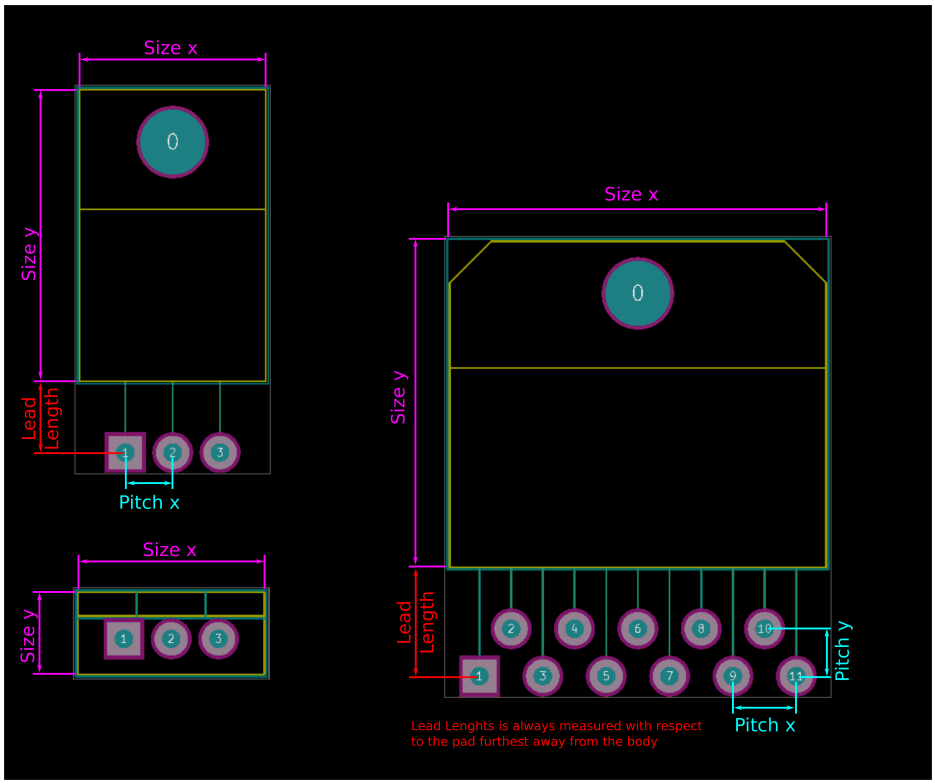
Notes:

1. Drill size (if specified) must be given in `mm`
2. `Orientation` can be chosen from:

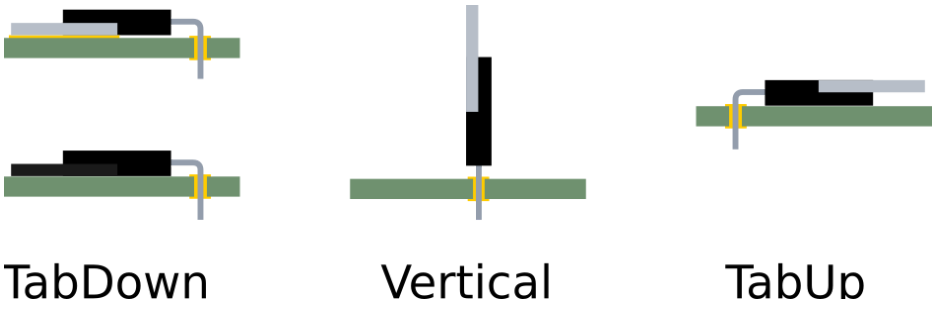
- Vertical
- TabDown
- TabUp

3. `Stagger` can be chosen from:
    - StaggerOdd (odd pins are further from component body)
    - StaggerEven (even pins are further from component body)

Size Parameter:



Orientation:



F3.6 Connector naming conventions

The following footprint naming conventions should be used as *examples* for naming connector footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields
- Mandatory fields
- Optional fields

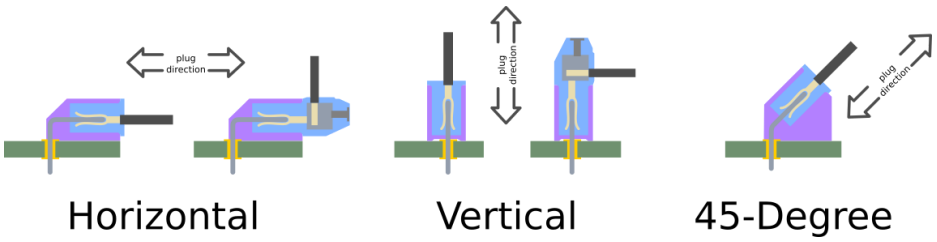
Conventions for Connectors

Connector footprints are extremely diverse and the naming conventions for connectors can be complicated. This section details some examples of naming schemes for various connector types. While there are variations between connectors, the general format for naming connectors is as follows:
[*Series*]_[*Subseries*]-[*Pins*]_[*MPN*]_[*Pin layout*]_**P**[*Pitch*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **Series** - Generic series name (Optional)
- **Subseries** - Series secondary identifier (Optional)
- **Pins** - Pin count (Optional)
- **MPN** - Manufacturer specific identifier (Optional)
- **Pin layout** - Pin configuration, specific to connector type (Optional)
- **Pitch** - Pin pitch, if not implied by connector standard (Optional)
- **Modifiers** - Modifiers to connector specifications (pad size, etc) (Optional)
- **Orientation** - Orientation of connector relative to board (Optional)
- **Options** - Extra footprint options (Optional)

Notes:

1. [`Series`] - a versatile field which could contain:
    - Functional description (e.g. `USB`)
    - Connector type (e.g. `PinHeader` or `TerminalBlock`)
    - Industry standard (e.g. `DSUB-15`)
    - Manufacturer specific series (e.g. `Hirose_DF13`)
2. [`MPN`] (manufacturer part number) - uniquely identifies a connector according to that manufacturer's internal naming conventions
3. [`Pin layout`] - dependent on the connector type
    - Single row connectors: 1x[`number of pins`]
    - Multi row connectors with equal number of pins in each row: [`number of rows`]x[`number of pins per row`]
    - Other connectors: [`number of rows`]Rows_[`number of pins overall`]Pins
    - Connectors for standardized interfaces do not need to include the pin configuration information (e.g. USB)
4. Number format:
    - [`number of rows`] number of rows no leading zero.
    - [`number of pins per row`] and [`number of pins`] two digit number with leading zero. (can have more digits for connectors with > 99 pins)
5. [`Pitch`] - if the connector standard does not imply the pin pitch, then this field is required
6. [`Modifiers`] - multiple fields may be added here, detailing modifications or devations from the standard footprint design. Examples include:
    - Pad size variation
    - Drill size variation
7. [`Orientation`] - The orientation of the connector with reference to the board plane



7. [`Options`] - Extra footprint options

In addition to these conventions, additional schemes are employed for specific connector types - these are detailed below.

Connectors for Specific Functions

Many connectors are designed for specific functions (e.g. USB, HDMI, SD-Card, etc). In these cases, specific naming conventions should be followed:

[*Function*]_[*Standard*]_[*Size*]-[*Type*]_[*MAN*]_[*MPN*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **Function** - Functional purpose of the connector
- **Standard** - Standard to which the connector complies, if applicable (Optional)
- **Size** - Connector size (e.g. Micro/Mini) (Optional)
- **Type** - Connector sub-type (e.g. A/B/C) (Optional)
- **MAN** - Manufacturer name (Optional)
- **MPN** - Manufacturer part number (Optional)
- **Modifiers** - Footprint modifiers (Optional)
- **Orientation** - Connector orientation (Optional)
- **Options** - Extra footprint options (Optional)

Example:

USB_Micro-B_Molex_105133-0001_Vertical

Notes:

1. Some *examples* of functional connector naming are provided below:
    - `USB_Micro-B_Wuerth_614105150721_Vertical_CircularHoles`

- `HDMI_Micro-D_Molex_46765-2x0x`
- `MicroSD_Wuerth_WR-CRD_693072010801`

2. In most cases, pin layout does not need to be explicitly specified

Connectors with Standardised Shape

Many connectors are provided by multiple manufacturers in an industry standard package or physical layout. These connectors should use the accepted industry naming at the beginning of the footprint name:

[*Standard*]-[*Pins*]_[*Series modifier*]_[*MAN*]_[*Series*]_[*MPN*]_[*Pin layout*]_**P**[*Pitch*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **Standard** - Generic connector standard (see notes below)
- **Pins** - Number of pins
- **Series modifier** - Series Modifier (see notes below)
- **MAN** - Manufacturer name (Optional)
- **Series** - Manufacturer series name (Optional)
- **MPN** - Manufacturer part number (Optional)
- **Pin layout** - Pin configuration, if not fully defined by applicable standard (Optional)
- **Pitch** - Pin pitch, if not fully defined by applicable standard (Optional)
- **Modifiers** - Footprint modifier field(s) (Optional)
- **Orientation** - Connector orientation (if multiple options exist) (Optional)
- **Options** - Extra footprint options (Optional)

Example:

DSUB-15[_][HighDensity][_][Vertical]

Notes:

1. The generic connector type [`Standard`] defines the shape of the connector

- e.g. `DSUB Tab Circular`

Generic Connector Types

Examples of generic arrayed connectors:

- PinHeader
- PinSocket
- TerminalBlock

[*Connector type*]_[*MPN*]_[*Pin layout*]_**P**[*Pitch*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **Connector type** - Generic connector style (see notes below)
- **MPN** - Manufacturer specific identifier (Optional)
- **Pin layout** - Pin configuration layout (see notes below)
- **Pitch** - Pin pitch
- **Modifiers** - Modifiers to connector specifications (pad size, etc) (Optional)
- **Orientation** - Orientation of connector relative to board
- **Options** - Extra footprint options (Optional)

Example:

PinHeader_1x10_P2.54mm_Drill1.25mm_Horizontal

Manufacturer Specific Connectors

Manufacturer series connectors are those which are particular to a particular manufacturer standard.

[*MAN*]_[*Series*]_[*MPN*]_[*Pin layout*]_**P**[*Pitch*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **MAN** - Manufacturer name (Optional)
- **Series** - Manufacturer series name
- **MPN** - Manufacturer part number
- **Pin layout** - Pin layout information (see notes below)
- **Pitch** - Pad pitch

- **Modifiers** - Modifiers to connector specifications (pad size, etc)
- **Orientation** - Direction of connector insertion relative to board plane
- **Options** - Extra footprint options (Optional)

Example:

JST_GH_S12B-EH_1x13_P2.50mm_Pad1.80mm_Horizontal

Notes:

1. If the footprint is located in a library dedicated to a single manufacturer, then the `MAN` prefix is not required

F3.7 Fuse naming conventions

The following footprint naming conventions should be used as *examples* for naming fuse and fuse holder footprints.

If you do not find an appropriate convention that matches a particular footprint type, either contact the KiCad library team or try to match a convention set by existing library components.

In the entries below, variable fields are denoted as follows:

- Fixed fields
- Mandatory fields
- Optional fields

Fuse Holders

FuseHolder_[*Fuse Type*]_[*MAN*]_[*Series*]_[*MPN*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **Fuse Type** - Type of fuse (Optional)
- **MAN** - Manufacturer name (Optional)
- **Series** - Manufacturer series name (Optional)
- **MPN** - Manufacturer part number (Optional)

- **Modifiers** - Footprint modifiers if non-standard (Optional)
- **Orientation** - Orientation of fuse, if multiple orientations are available (Optional)
- **Options** - Extra footprint options (Optional)

Example:

FuseHolder_Littelfuse_TE5_395

Examples of `Fuse Type` field:

- Automotive fuses:
  - Automotive-Micro2
  - Automotive-Micro3
  - Automotive-LowProfile
  - Automotive-Mini
  - Automotive-ATO
  - Automotive-Maxi
  - Automotive-Bosch
  - Automotive-SFE-[SizeCode]
- Industrial fuses
  - Cylinder-[diameter]x[length]mm

Fuses

Cylindrical Fuse

Fuse_[*MAN*]_[*Series*]_[*MPN*]_[*Body size*]_[*Modifiers*]_[*Orientation*]_[*Options*]

- **MAN** - Manufacturer name (Optional)
- **Series** - Manufacturer series name (Optional)
- **MPN** - Manufacturer part number (Optional)
- **Body size** - Major fuse dimensions
- **Modifiers** - Footprint modifiers if non standard (Optional)
- **Orientation** - Orientation of fuse, if multiple orientations are available (Optional)
- **Options** - Extra footprint options (Optional)

Example:

Fuse_Littelfuse_LVR100

Rectangular SMD Fuses

Fuse_[*Imperial size*]_[*Metric size*]**Metric**_[*Modifiers*]_[*Orientation*]_[*Options*]

- **Imperial size** - Imperial case size (Optional)
- **Metric size** - Metric case size (Optional)
- **Modifiers** - Footprint modifiers if non standard (Optional)
- **Orientation** - Orientation with respect to board plane (Optional)
- **Options** - Footprint options (Optional)

Example:

Fuse_0805

F4 - General Footprint Requirements

F4.1 Datasheet recommendations take priority

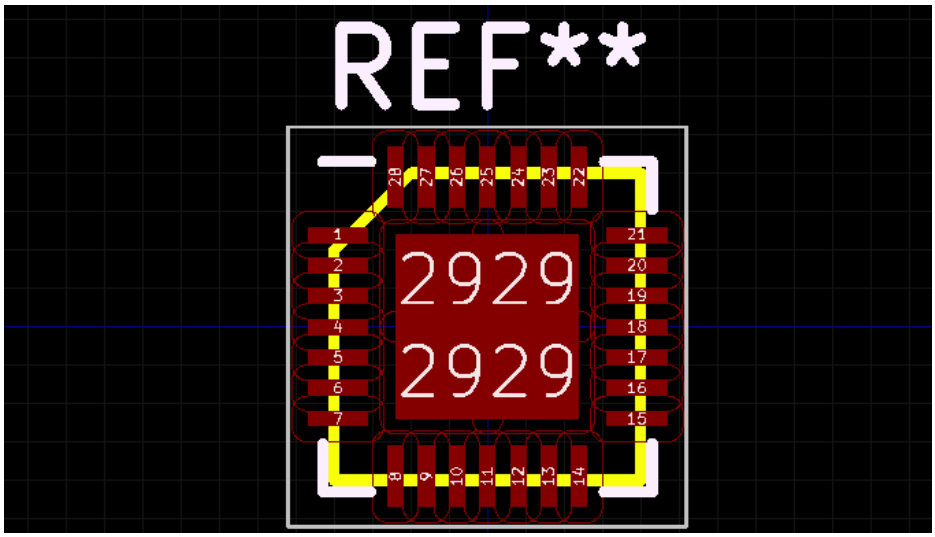
Where conflict exists between datasheet recommendations and KLC requirements, then (in the general case) the datasheet should take priority.

It is important when designing footprints that they adhere to requirements for soldering and assembly - often these requirements have very small tolerance for variation.

If the KLC is at odds with the datasheet, follow the datasheet, and note this difference to the KiCad library team when submitting your contribution.

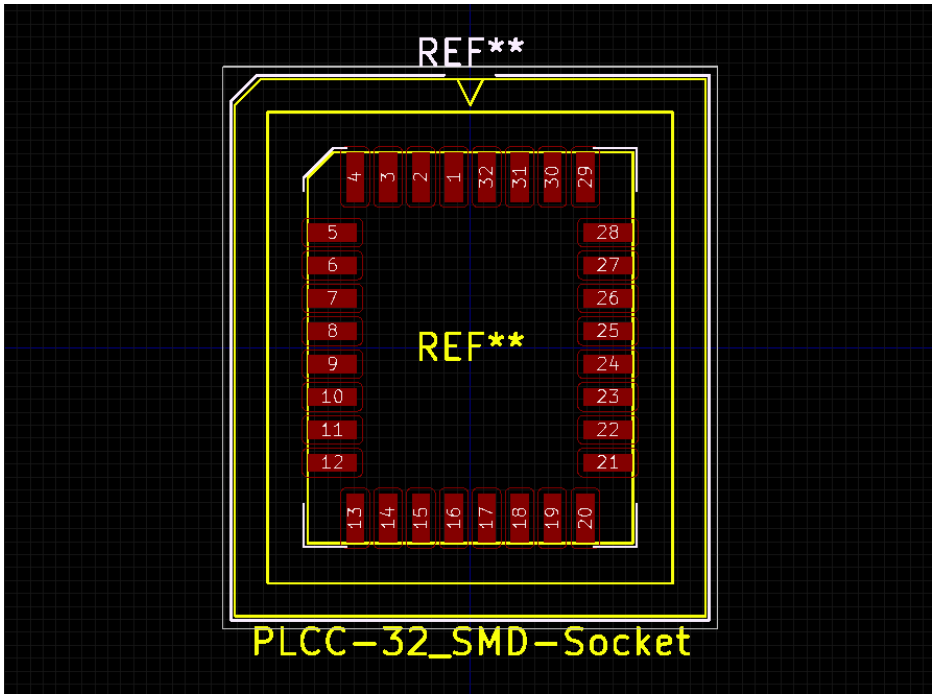
F4.2 Pin 1 should be located at the top left

Footprints should be oriented such that Pin 1 is located in the upper left corner (IPC-7351).

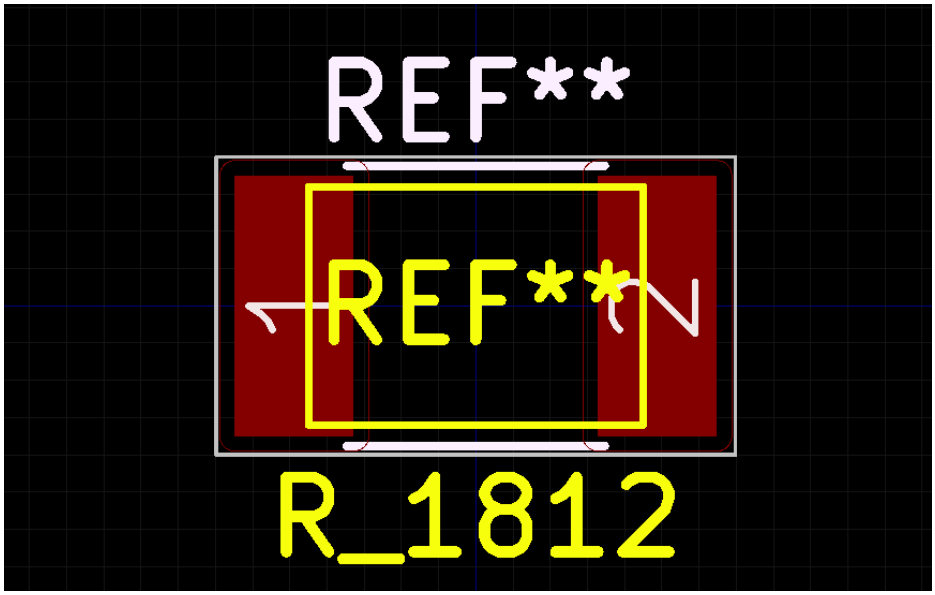


Exceptions

Where footprints cannot be oriented with pin 1 in the top-left quadrant, pin 1 should be aligned to the top



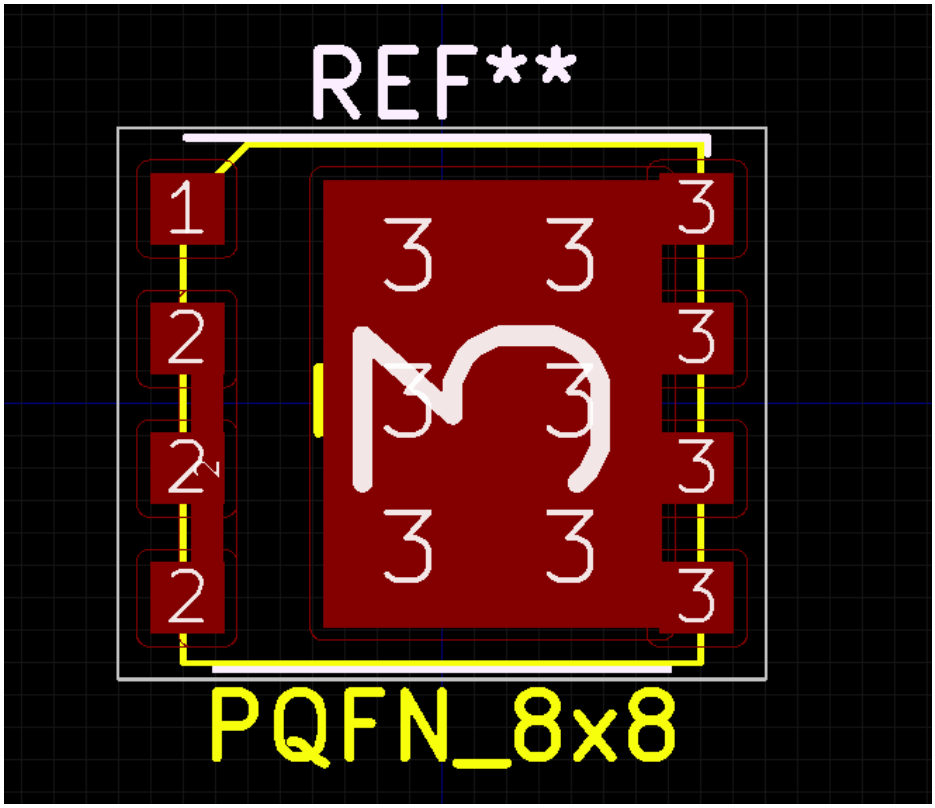
Two terminal footprints should be aligned with pin 1 on the left side



F4.3 Connected copper elements have the same pad number

Footprints that contain multiple pads or conductive elements that are **physically connected** require special attention:

1. Multiple pads that are physically connected must share the same number
2. Thermal vias (if present) must share the same number as the thermal pad to which they are connected



F4.4 Thermal pads

Manufacturer datasheets may specify that thermal vias should be added to the thermal pad under a device package, to assist in moving heat away from the device.

In such cases the datasheet will usually specify the quantity, size and position of these vias.

In addition to manufacturer requirements, a number of extra conventions should be followed when creating thermal vias.

1. Thermal vias must share the same pad number as the pad on which they are placed
2. Where thermal vias are connected to a pad on the `F.Cu` layer, a copper pad on the `B.Cu` layer should be added, to provide thermal relief for the via. `B.Mask` is not to be enabled on the `B.Cu` pad so mask is present. This pad must be large enough to fully surround the via(s) with a margin of `>= 0.5mm`. *Note: If the datasheet gives conflicting advice to the placement of*

this pad on the `B.Cu` layer, then the datasheet instruction should be given preference.

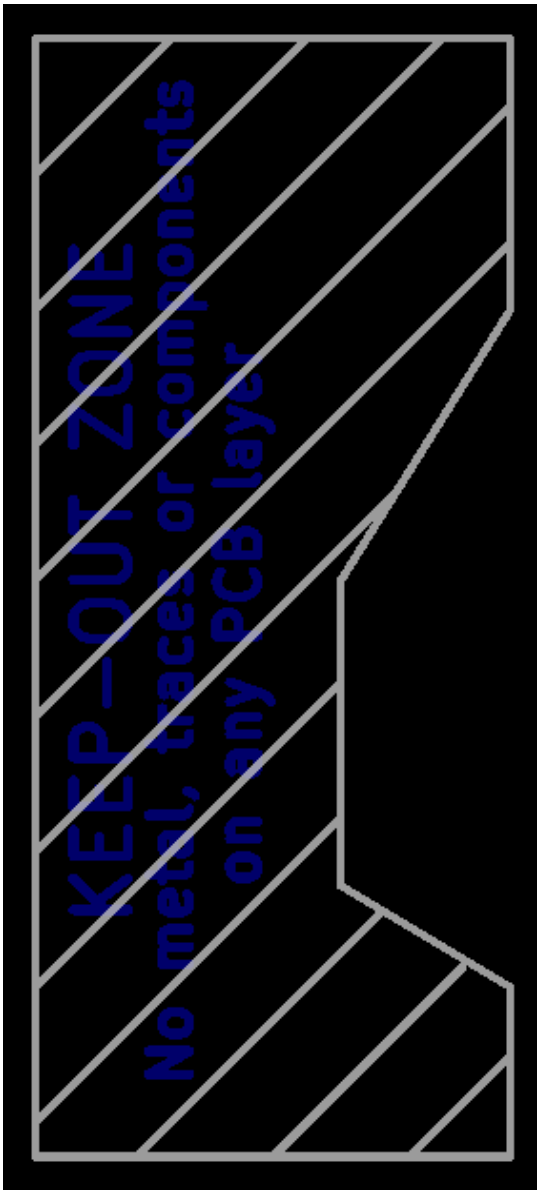
3. Unless instructed by the datasheet, solder paste must not be placed over the thermal via(s). This can cause solder to wick into the via during reflow, and cause solder joint issues.
4. When submitting a footprint with thermal vias, the suffix `_ThermalVias` should be added to the footprint name
5. If it does not already exist, a second version of the footprint without the thermal vias should be added. *Note: This second version should not have the `_ThermalVias` suffix in the footprint name.*

F4.5 Specifying footprint keepout areas

KiCad does not yet have a method for drawing keepout areas inside footprint files. Until this feature is implemented, footprint keepout must should be indicated using the following procedure:

1. Keepout area shape should be drawn on the `Dwgs.User` layer
2. Area should be hatched diagonally
3. Descriptive text may be included on the `Cmts.User` layer. This text is used to provide further information about keepout requirements to the user.

Example:



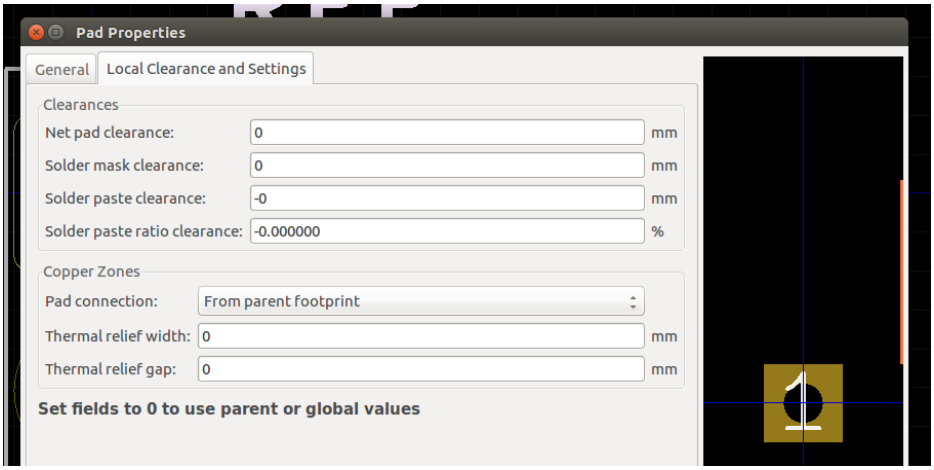
F4.6 Local clearance and settings should be set to zero

Pad clearance values are determined using a priority system, which checks the following locations for clearance values, in descending order of priority. If a particular setting reads as zero (`0`) then it is ignored, and the next location is checked.

1. **Local pad settings** - If an individual pad specifies clearance values, these are used
2. **Footprint settings** - If clearance values are specified in the **Footprint Properties** window, these are used

3. **Global settings** - If no values are specified as above, then the global values (as specified in PCB settings dialog) are used.

Unless there is a specific reason for setting explicit values for local clearance values for pads or pins, they should be left at zero (`0`).



Exceptions:

If the component datasheet calls for specific clearance values for a particular pad (or the entire footprint) then these values should be used as appropriate.

F5 - Layer Requirements

F5.1 Silkscreen layer requirements

The silkscreen is printed to the external surface of a PCB to aid in component identification and orientation. Typically this layer contains the component RefDes to locate components on the board after assembly.

KiCad refers to the silkscreen layers as:

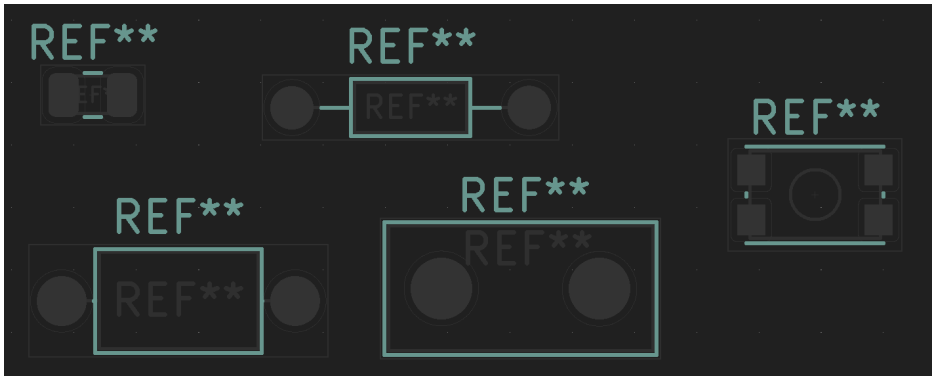
- `F.SilkS` - Front silkscreen layer
- `B.SilkS` - Back silkscreen layer

The following elements must be provided on the silkscreen.

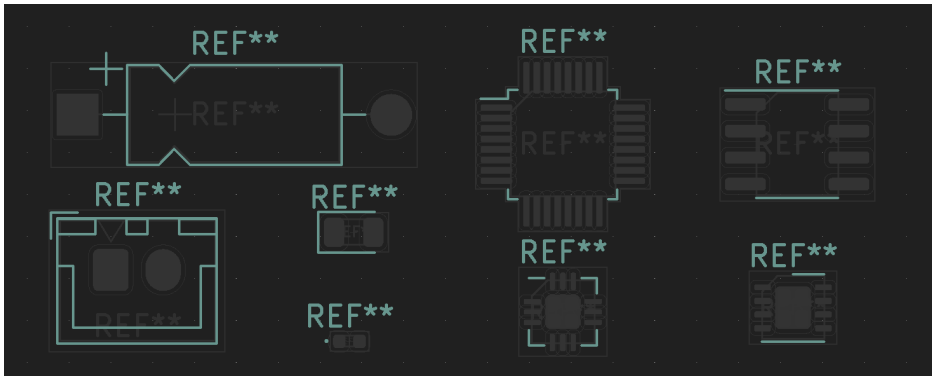
1. Reference Designator must be drawn on `F.SilkS` layer
    a. Text size = `1.00mm`
    b. Text thickness = `0.15mm`

2. Silkscreen line width is between {`0.10mm` and `0.15mm`} as per IPC-7351C:
    a. Silkscreen line width should nominally be `0.12mm`
    b. `0.1mm` is allowed for high density designs
    c. `0.15mm` is allowed for low density designs
3. Silkscreen must not be placed over pads or areas of exposed copper
    a. Clearance between silkscreen and exposed copper elements is recommended to be 0.2mm.
    b. Clearance must be at least the silkscreen line width or pad mask expansion, whichever is greater.
4. For SMD footprints, silkscreen must be fully visible after boards assembly (no silkscreen allowed under component)
5. For THT components, additional silkscreen may be placed under component to aid in assembly process
6. Pin-1 designator is provided on the `F.SilkS` layer
7. Pin-1 designator must be visible after board assembly

Examples of silk for non polarized components



Examples of silk for polarized components



F5.2 Fabrication layer requirements

The fabrication layers are used to display the simplified mechanical outline of components on the PCB.

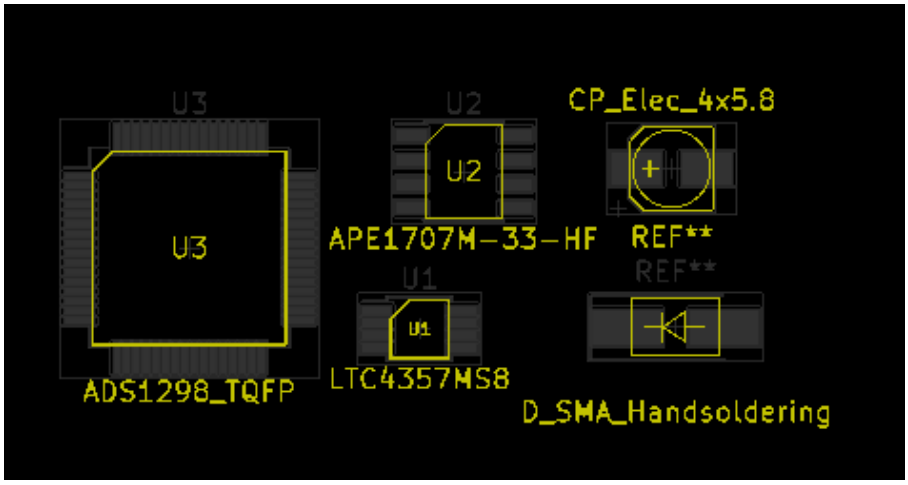
KiCad refers to the fabrication layers as:

- `F.Fab` - Front fabrication layer
- `B.Fab` - Back fabrication layer

The following elements must be provided on the fabrication layer(s)

1. Simplified component outline must be provided on `F.Fab` layer
    a. Outline uses line width between {`0.10mm` and `0.15mm`} (recommended `0.10mm`)
    b. Outline should be simplified and not display complex features
    c. For outlines based on the component body shape, the nominal size is used
2. Footprint polarisation / location of pin-1 is drawn
    a. For IC packages, bevel is drawn at corner next to pin-1
    b. Bevel should be `1mm` or `25% of package size` (whichever is smaller)
    c. For connectors, a small arrow indicator drawn next to pin-1 should be used
3. Component value (footprint name) must be displayed on the `F.Fab` layer
    a. Recommended text size = `1.0mm`
    b. Allowable text size = {`0.5mm` to `1.0mm`}
    c. Text thickness should be approximately `15%` of text size (with allowances for variation for aesthetic reasons)
    d. Placed below (positive y direction) the part outline.
4. A second copy of the reference designator (RefDes) must be provided on the `F.Fab` layer. To add a second RefDes item, add a text object with the value `%R`

a. RefDes must be centered on component body (inside component outline)
b. Orientation of RefDes should match major component axis
c. Size of text should be scaled to match component size
    i. It is recommended to scale it such that 4 characters fit without overlapping other features of the same layer.
    ii. If it is not possible to fit at least 3 characters with the text size restrictions, then the reference should be moved outside (but scaled to smallest allowable text size)
d. Recommended text size = `1.00mm`
e. Allowable text size = {`0.5mm` to `1.0mm`}
f. Text thickness should be approximately `15%` of text size (with allowances for variation for aesthetic reasons)



F5.3 Courtyard layer requirements

The component courtyard is defined as the smallest rectangular area that provides a minimum electrical and mechanical clearance around the combined component body and land pattern boundaries. It is allowed to create a contoured courtyard area using a polygon instead of a simple rectangle. (IPC-7351C)

The courtyard should include any extra required clearance for mating connectors (for example).

KiCad refers to the courtyard layers as:

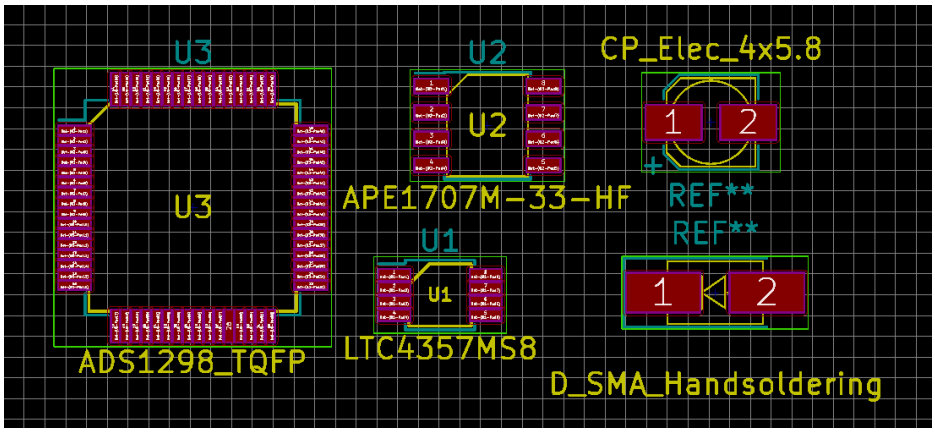
- `F.CrtYd` - Front courtyard layer
- `B.CrtYd` - Back courtyard layer

A fully enclosed Component courtyard must be drawn on the `F.CrtYd` layer, with the following parameters:

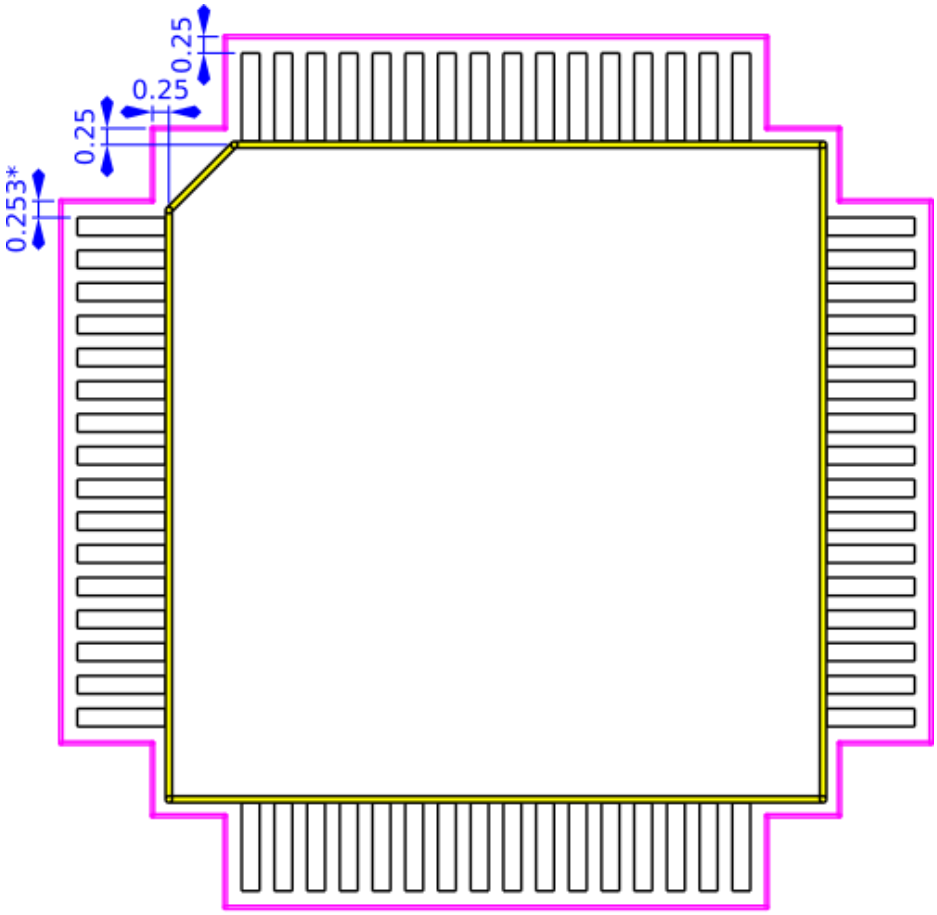
1. Courtyard uses `0.05mm` line width
2. All courtyard line elements are placed on a `0.01mm` grid
3. If the component requires a courtyard on the back of the PCB, a corresponding courtyard must be provided on the `B.CrtYd` layer.
4. Where the courtyard depends on the dimensions of the physical part body, clearance is calculated from the nominal part dimensions.

Courtyard clearance should adhere to the following requirements:

5. Unless otherwise specified, clearance is `0.25mm`
6. Components smaller than `0603` should have a clearance of `0.15mm`
7. Connectors should have a clearance of `0.5mm`, in addition to the clearance required for mating of connector
8. Canned capacitors should have a clearance of `0.5mm`
9. Crystals should have a clearance of `0.5mm`
10. BGA devices should have a clearance of `1.0mm`



Example for courtyard clearance when applied to the contoured courtyard outline of a QFP-64 package. (Shown are courtyard, fab-outline and copper pads.)



F5.4 Elements on the graphic layer should not overlap

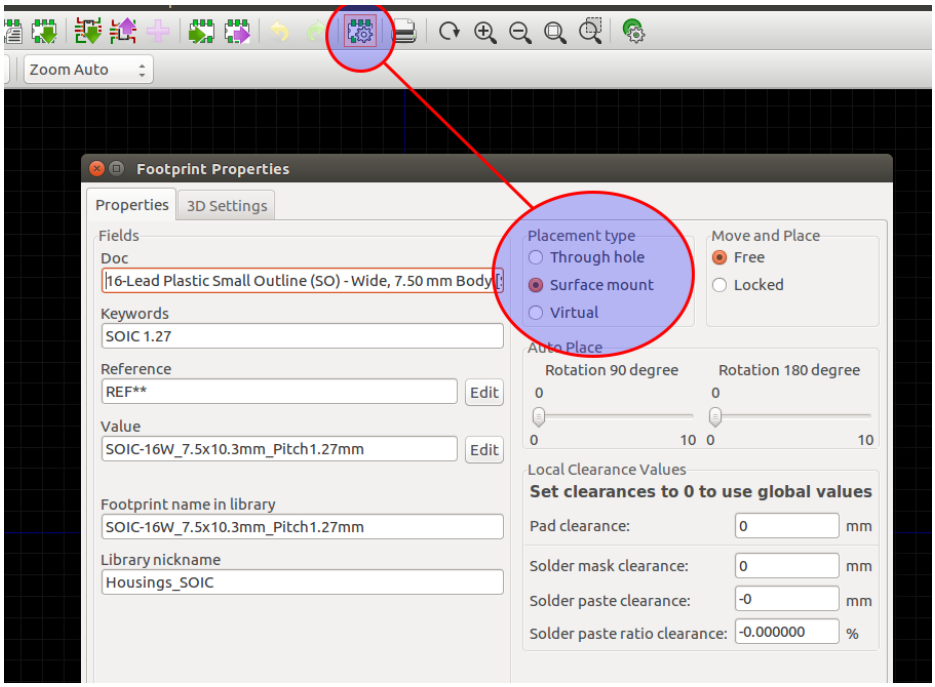
Footprints should not have overlapping or duplicate graphic elements. This includes lines, arcs and circles. Touching endpoints or intersections are not considered as overlapping.

F6 - Surface Mount Components

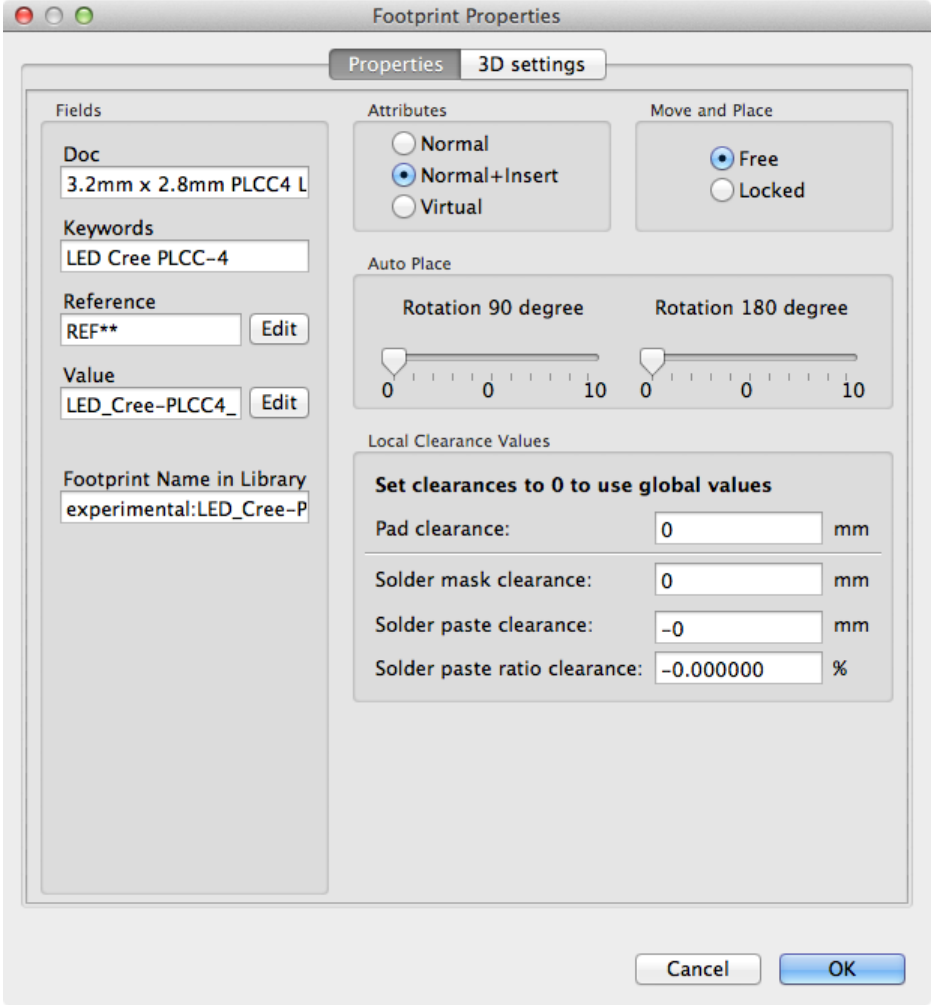
F6.1 Footprint placement type must be set to surface mount

The **Footprint placement type** must be set to `Surface mount` for SMD footprints. This is to ensure that these footprints are included in the **Footprint position (.pos)** file output.

To set the footprint placement type, open the **Footprint properties** window and select **Surface Mount** as indicated.



In KiCad 4.x, **Placement type** is called **Attributes** and **Surface mount** is called **Normal+Insert**.

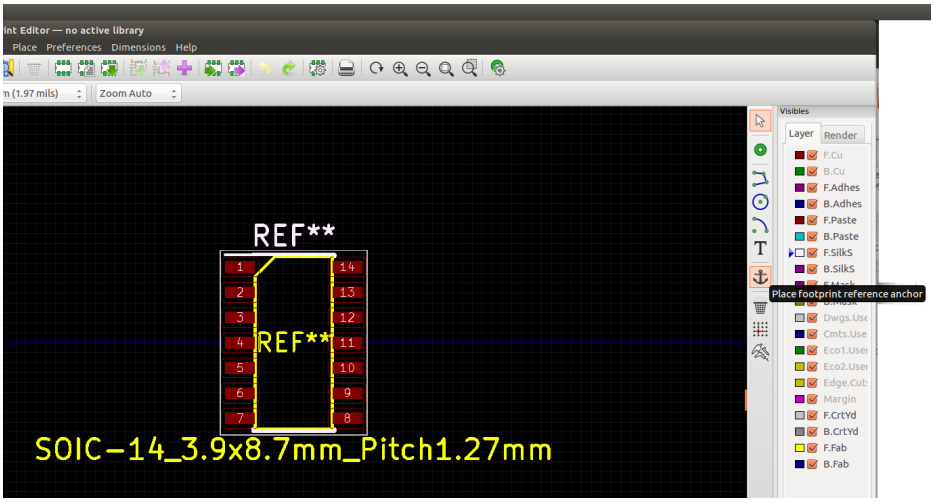


F6.2 Footprint anchor should be placed in the middle of the component body

The footprint anchor (also called *component origin*) is used by automated Pick and Place (PNP) machines for locating and placing SMD components on a PCB.

For most standard components, the anchor should generally be located on the centroid of the component body.

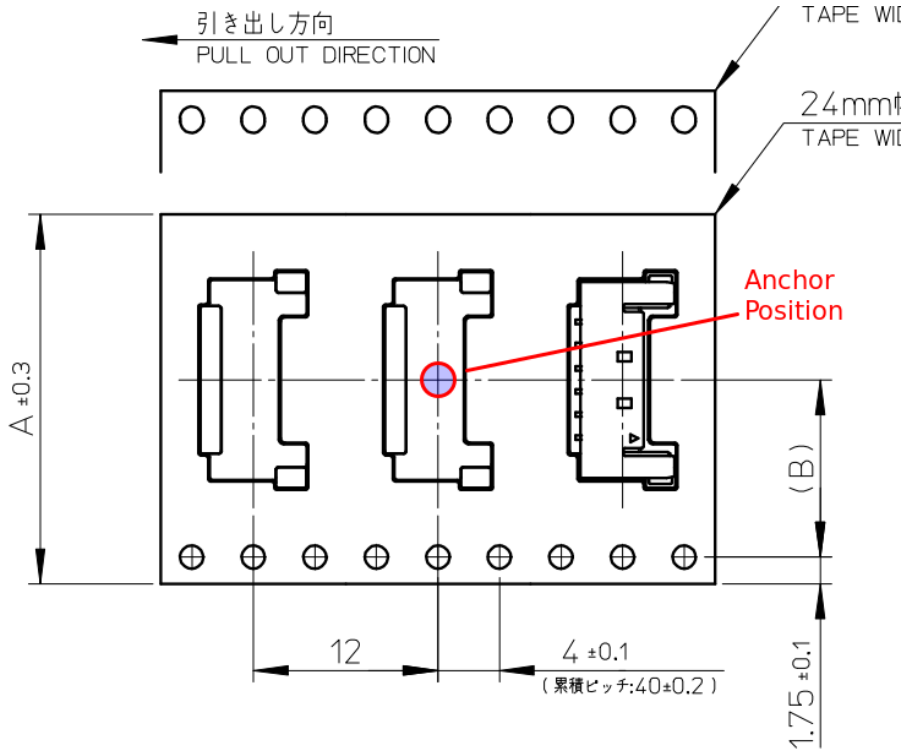
In the KiCad footprint editor, the footprint anchor is *always located at the origin* (0, 0). To move the footprint relative to the anchor (origin), press the **Place footprint reference anchor** button (refer to image below).



Exceptions:

Some footprints (especially those which are non-symmetrical) require special consideration and the anchor may not necessarily be placed on the component centroid.

Generally speaking, the correct location of the footprint anchor can be found in the component datasheet. The image below shows an example for a **Molex Pico-Lock** connector.



F6.3 Pad requirements for SMD footprints

Surface mount pads have specific requirements for PCB design.

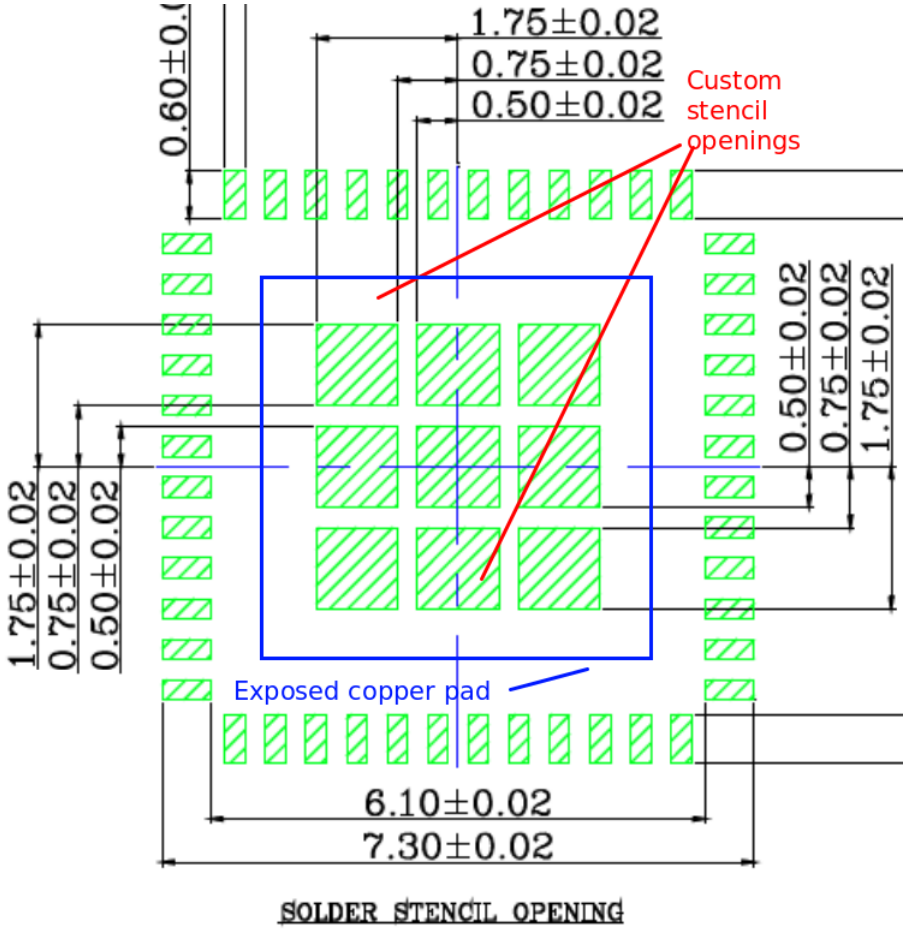
1. By default, pads for SMD footprints should use only the following layers
    a. `F.Cu` - Front copper
    b. `F.Mask` - Front soldermask
    c. `F.Paste` - Front solderpaste (stencil openings)
2. If SMD pads are placed on the back of the PCB, then the following layers should be used:
    a. `B.Cu` - Back copper
    b. `B.Mask` - Back soldermask
    c. `B.Paste` - Back solderpaste (stencil openings)
3. Pads with specific stencil aperture design requirements require special attention (see details below)

Requirements for solderpaste

Many components (IC packages in particular) have specific design requirements for solderpaste (stencil) design. These requirements are often found in the footprint datasheet or an associated technical document.

Most often, custom stencil openings are required for the exposed pads on large IC packages to reduce the amount of solderpaste that is applied during stencil application.

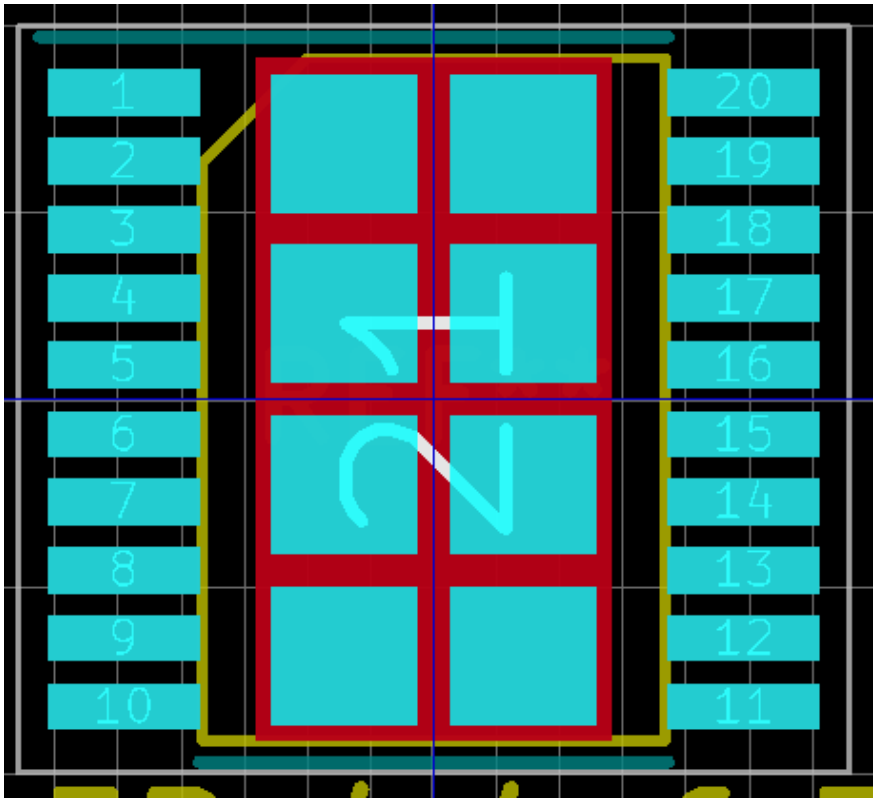
The image below shows an example stencil design found in a footprint datasheet. The superimposed blue square represents the shape of the exposed copper pad. The green squares are the required openings in the stencil.

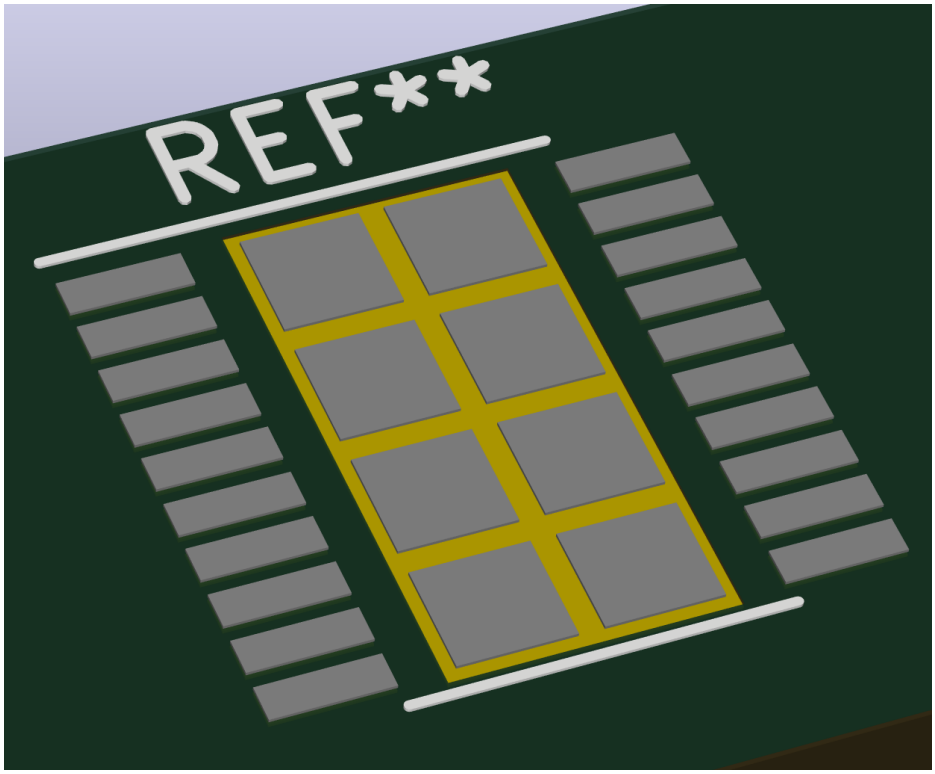


Custom stencil openings are accommodated as follows:

a. Construct copper shape with pad(s) with only copper layers (`F.Cu` and/or `B.Cu`) as appropriate. These copper pads should not have the `F.Paste` or `B.Paste` layers checked.
b. Add stencil opening(s) with pad(s) which only use solderpaste layers (`F.Paste` and/or `B.Paste`) as required. These pads *do not have a pad number*.
c. The solderpaste should cover between 50% and 80% of the soldermask free pad area. (recommendation: 65%)

Result: *(note: this footprint does not correspond to the example shown above)*





Size of exposed pads

The following rules apply for finding the correct pad size for parts including an exposed pad.

1. By default the nominal size of the package pad is used.

2. Variation within the tolerance ranges of said package pad are allowed if the manufacturers suggested footprint suggest it that way.
    a. It is not allowed to use a footprint pad smaller than the minimum size of the package pad.
    b. Soldermask defined pas are to be used if the suggested footprint uses a pad larger than the maximum size of the package pad. (Meaning: copper can cover a larger area, but the exposed copper is still within the package pads tolerance range.)

Clearance requirement between exposed pads and normal pads

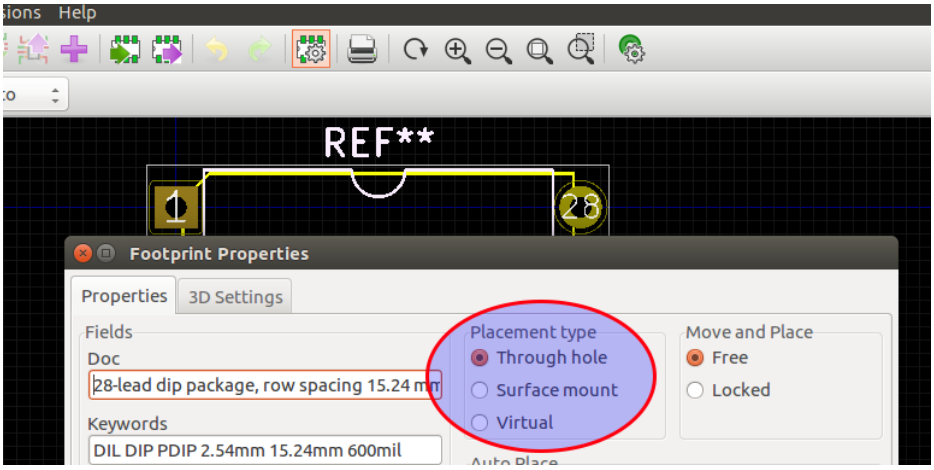
1. There should be clearance of at least 0.2mm between the exposed pad and the normal pads.
2. This clearance applies only to the area free of soldermask. (The clearance for soldermask defined pads is the spacing between the normal pads and the soldermask cutout of the exposed pad.)
3. The heel fillet of the normal pads is to be decreased if necessary to achieve this clearance.

F7 - Through Hole Components

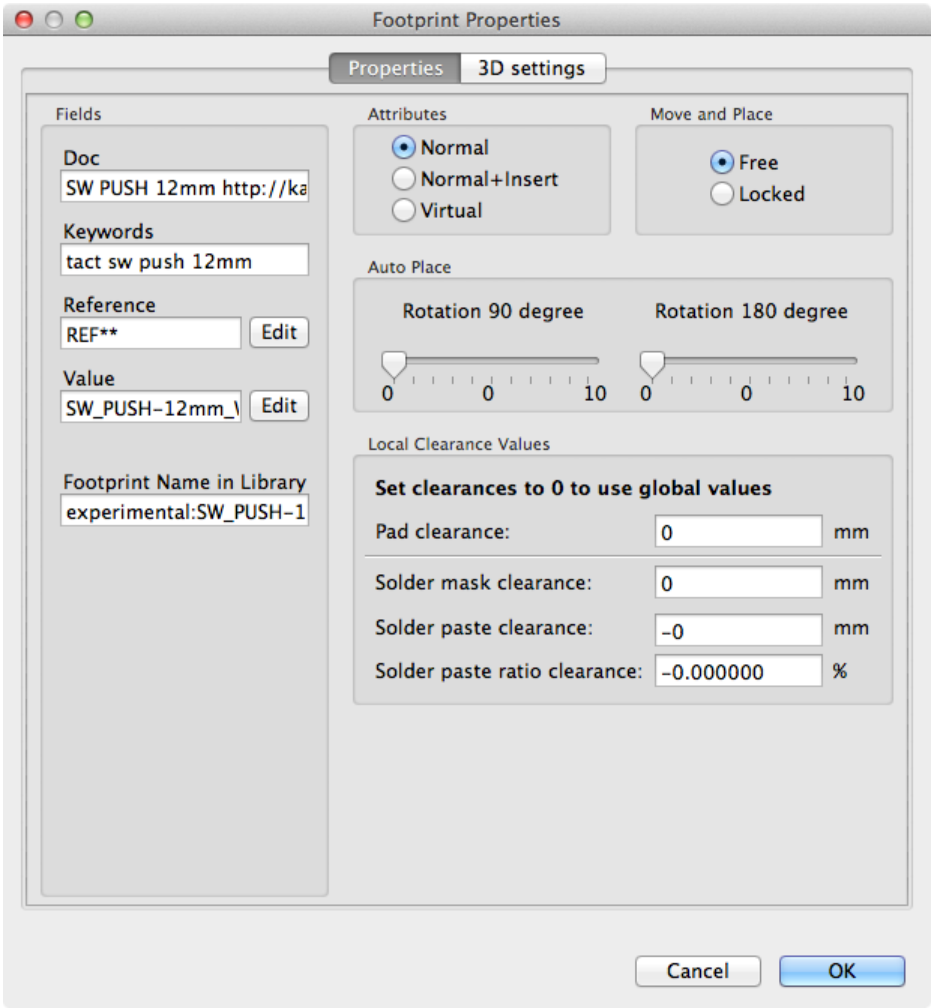
F7.1 Footprint placement type must be set to Through Hole

The footprint **placement type** must be set to `Through Hole` for THT footprints. This ensures that these footprints are not included in the Pick and Place (PNP) location files.

This can be selected in the **Footprint Properties** window

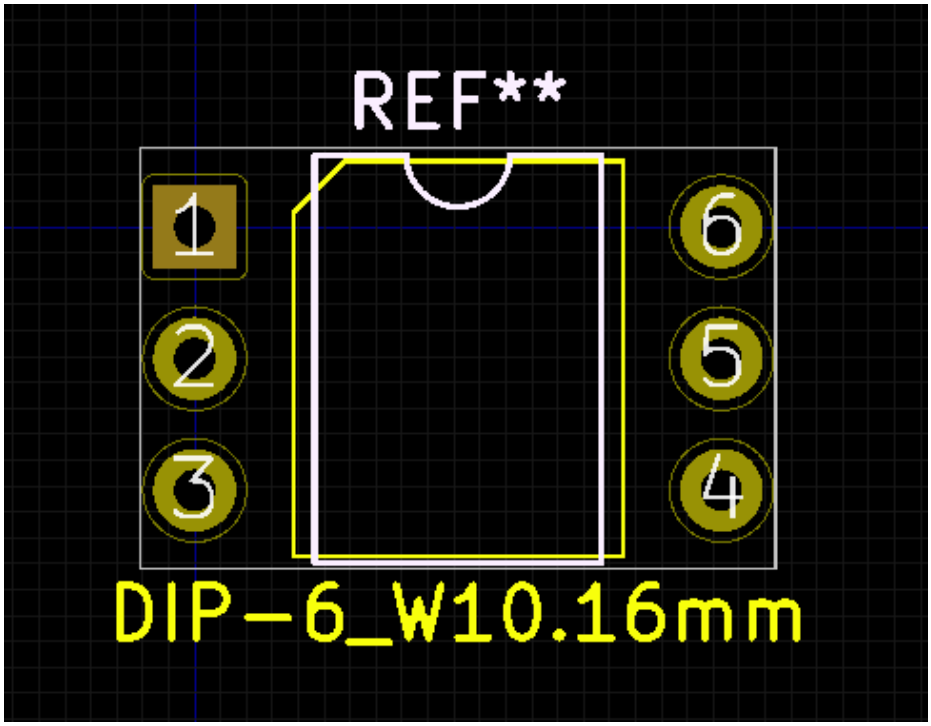


In KiCad 4.x, **Placement type** is called **Attributes** and **Through hole** is called **Normal**.



F7.2 Footprint anchor should placed at the location of Pin-1

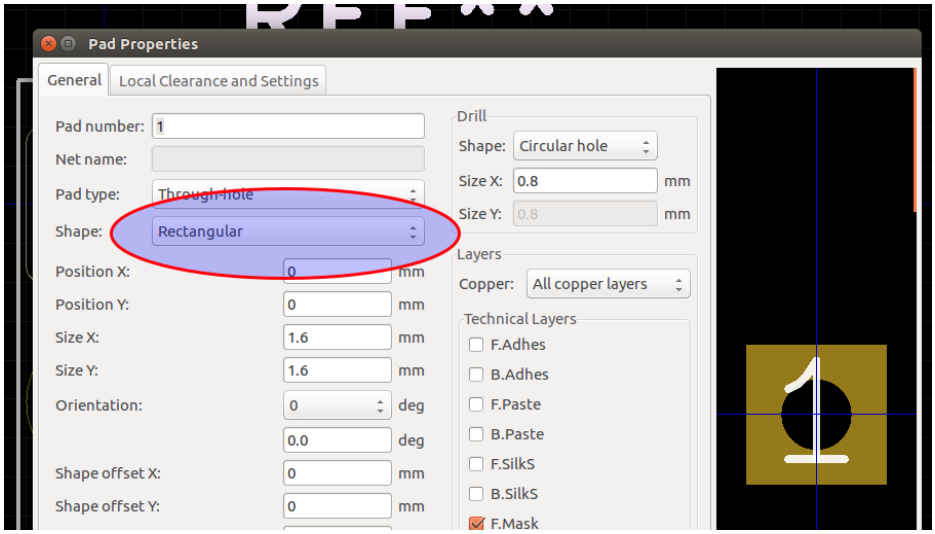
For through hole components the footprint anchor should be placed at the location of pin 1.



F7.3 Pin 1 should be rectangular or a rounded rectangle. Other pads circular or oval

Through hole components should set the shape of Pin 1 to `Rectangular` or `Rounded Rectangle`, and all other pads to either `Circular` or `Oval`. This aids in orienting the component during placement and also for locating Pin 1 for circuit troubleshooting. For rounded rectangle pads the corner size should be 25% with a maximum radius of 0.25mm.

Pad shape can be adjusted in the **Pad Properties** dialog:



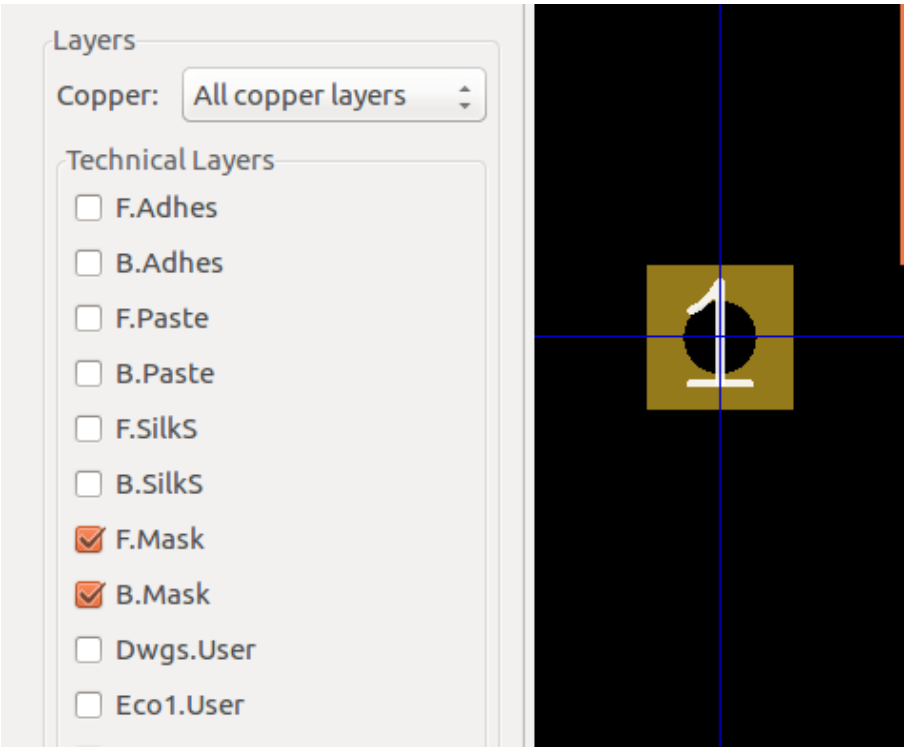
Exception:

Non polarized parts (such as THT resistors) should not set the shape of Pin 1 to Rectangular. For these (non polarized) footprints, all pad shapes should be consistent.

F7.4 Pad requirements for THT footprints

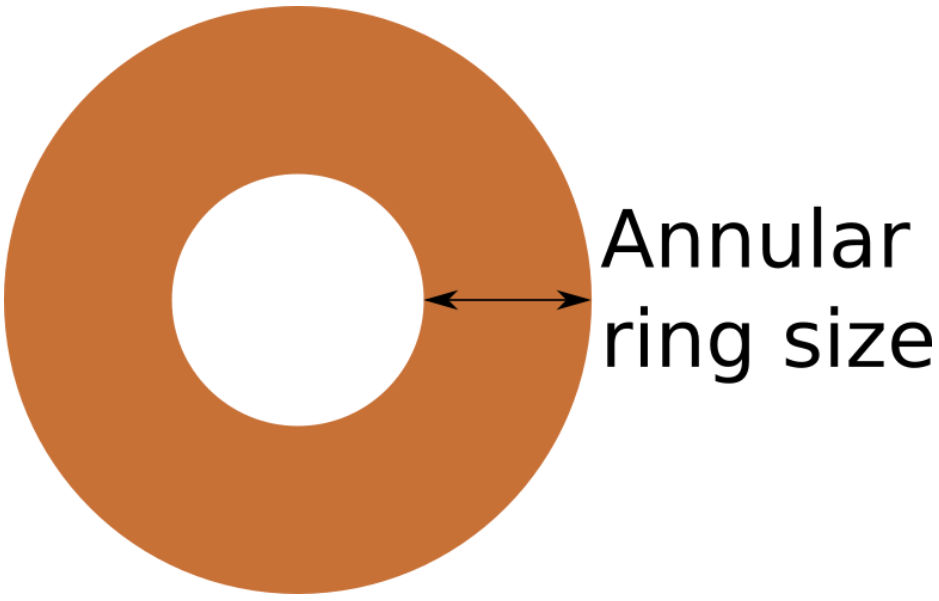
Through hole pads must have soldermask pulled back to allow soldering. However they must not have solderpaste openings as they do not receive solder paste during stencil screening.

1. Pads for THT footprints must have the following layers set:
    a. All copper layers
    b. `F.Mask` (front mask)
    c. `B.Mask` (back mask)
2. Pads must **not** have the silkscreen layers active



F7.5 Minimum annular ring width

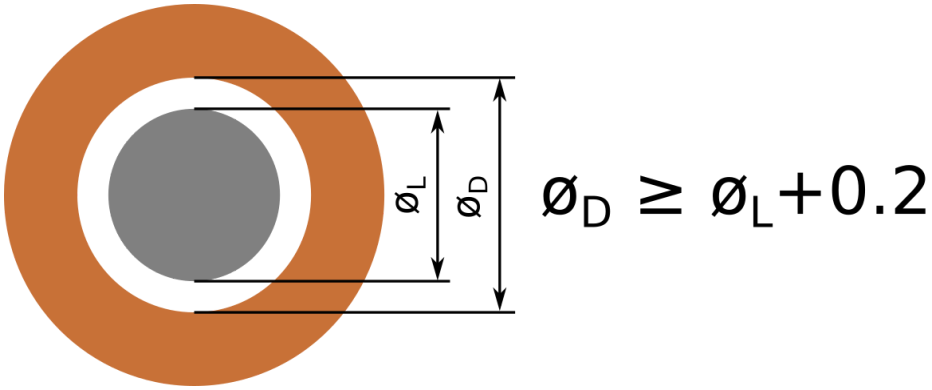
For drilled through-holes, the minimum annular ring width must be at least `0.15mm` (IPC-2221).



The annular ring is the copper pad which remains after the hole has been drilled.

F7.6 Minimum hole diameter

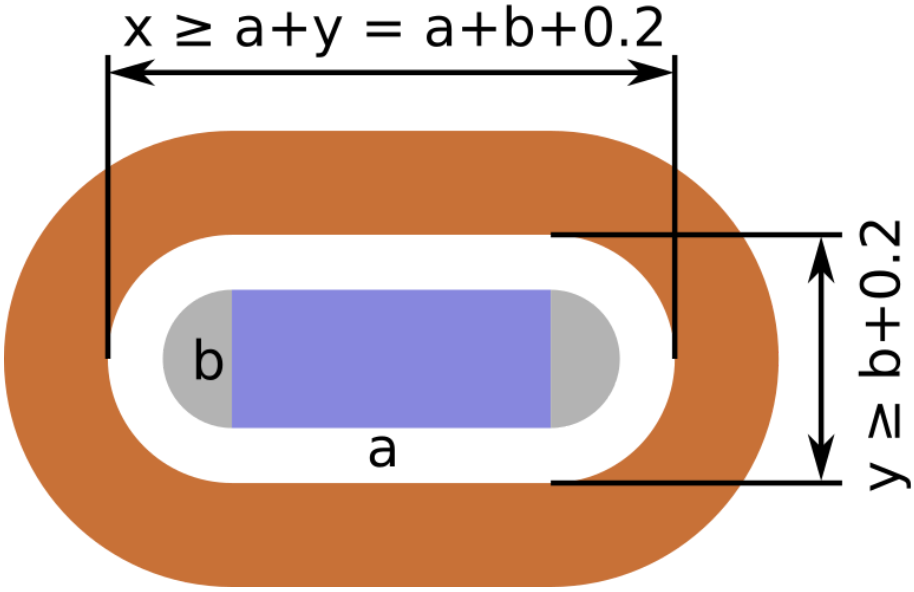
For drilled through-holes, minimum drilled hole diameter is the maximum lead diameter plus `0.2mm` (IPC-2222 class 2)



Maximum lead diameter is obtained from the device datasheet.

F7.7 Oval holes (plated milled slots)

The sizes for oval holes must be chosen such that there is 0.2mm clearance between the maximum lead size and the hole in all directions. Oval holes are only allowed if the lead length to width ratio is greater than 2. (Lead cross section is at least twice as long as wide)

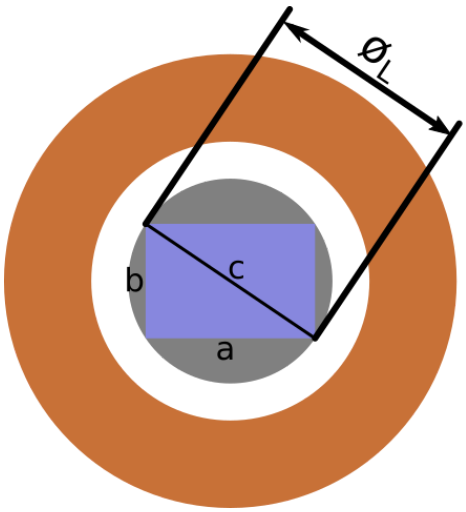


Maximum lead dimensions are obtained from the device datasheet.

Some PCB manufacturers are not able to produce milled slots. It is allowed to add an alternative footprint using only circular holes. These need to follow the same drill size increase as all other holes.

The equivalent lead diameter should be rounded up to the next 0.01mm increment.

The footprint name of such an alternative is marked with the suffix `_CircularHoles`.



$$c = \sqrt{(a^2+b^2)}$$
$$øL = \lceil c \rceil_{0.01}$$

F8 - Virtual Components

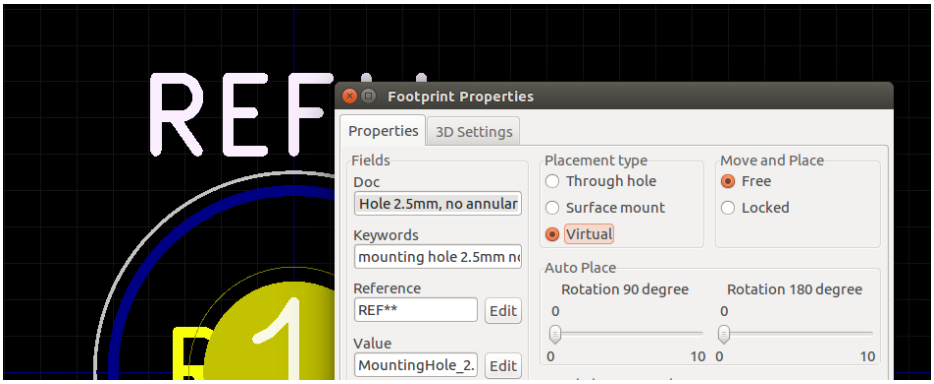
F8.1 Virtual components

Virtual components are those which have a footprint on the PCB (and may additionally have a schematic symbol) but do not have an associated physical component which needs to be loaded onto the board during assembly.

Examples of virtual components include:

- Mounting holes
- Solder bridges
- Net ties
- Test points
- Fiducial markings

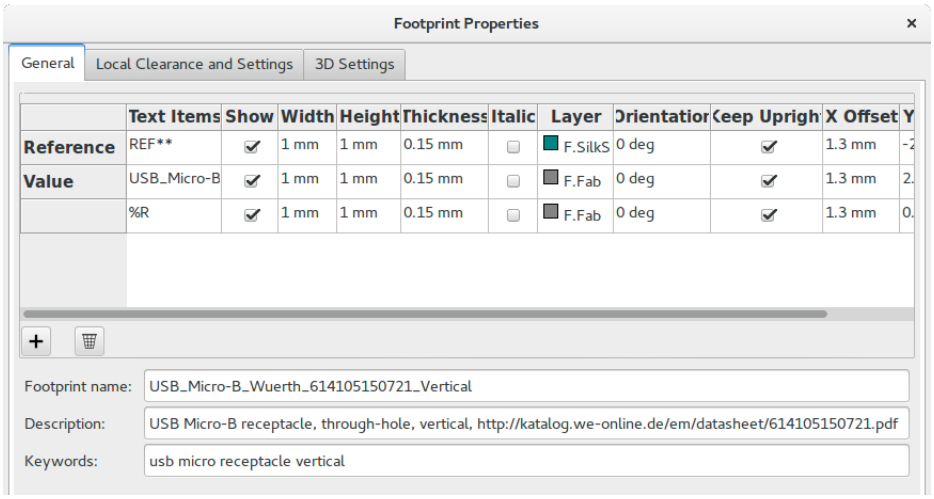
Virtual components must be indicated by setting the **Placement type** to `Virtual` in the **Footprint Properties** dialog.



F9 - Footprint Properties

F9.1 Footprint meta-data is filled in as appropriate

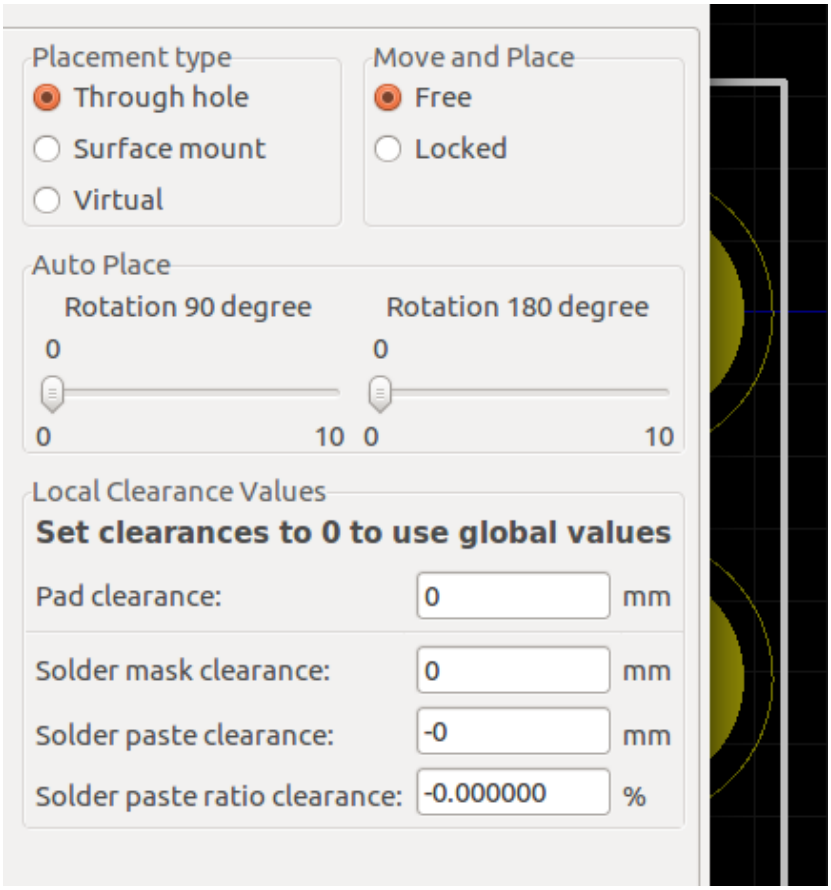
1. **Reference** field must be set to `REF**`
2. **Value** and **Footprint Name** fields must match footprint filename (ignoring the `.kicad_mod` extension)
3. **Description** field contains comma-separated device information. Where appropriate, a URL to the footprint datasheet should be included.
4. **Keywords** field contains space-separated keyword values



F9.2 Footprint properties are as default values unless otherwise required in datasheet

Footprint properties should be left as default, unless there is a reason to do so as specified by the footprint datasheet (for example).

1. **Move and Place** must be set to `Free`
2. **Auto Place** should be set to `0, 0`
3. **Local Clearance Values** should all be set to `0`



Exceptions:

The **Local Clearance Values** parameters are generally set to `0`, indicating that the global clearance values will be used to determine pad clearances.

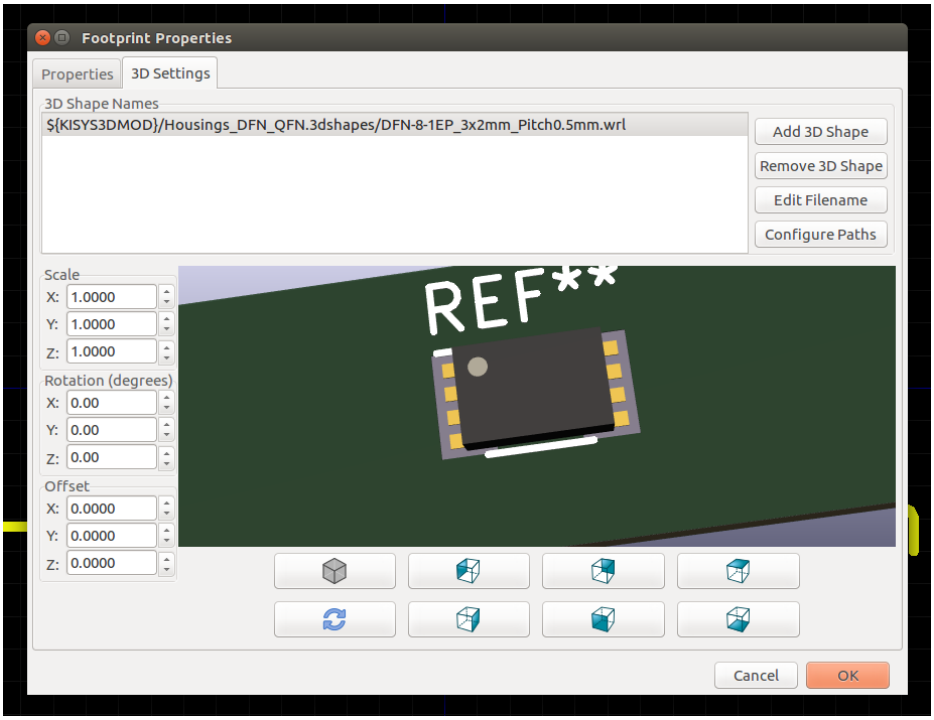
If *all* the pads on the particular footprint require a specific clearance value (for e.g. as described by the datasheet) then these values can be used here to override pad-specific settings for the entire footprint.

F9.3 Footprint 3D model requirements

1. All non-virtual footprints must have 3D model references, even if the 3D model is missing (does not yet exist). This allows the

3D model to be added later without requiring the footprint to be edited again. (It is not required to supply a 3d model file.)

2. 3D model files must be placed in a library (directory) which has the same name as the footprint library, with the extension `.3dshapes`
    - `Capacitor_SMD.pretty` → `Capacitor_SMD.3dshapes`
    - `Connector_USB.pretty` → `Connector_USB.3dshapes`
3. 3D model files should be named the same as the footprint (ignoring file extension)
    - `SOIC-8.kicad_mod` → `SOIC-8.wrl`
    - `R0805.kicad_mod` → `R0805.wrl`
4. If a footprint is a simple variation that does not change the 3D representation, the common 3D model should be used (*do not duplicate models unnecessarily*)
    - `R0805_HandSoldering.kicad_mod` → `R0805.wrl`
    - `QFN-48_ThermalVias.kicad_mod` → `QFN-48.wrl`
5. Model scaling must be 1:1:1 (model file should not require any scaling to correctly fit the footprint)
6. Rotation and offset must be 0 (The model must be aligned correctly to the footprint within the 3d modeling software.)
7. Path to 3D model must have the `${KICAD6_3DMODEL_DIR}/` prefix
8. Filetype must be `.wrl` (KiCad replaces it with .step automatically for the step export.)



3D Model Guidelines

The following guidelines apply to contribution of 3D model data.

M1 - Contributing Models

M1.2 The contributor must own the rights to share the 3D model

The goal of the KiCad 3d model library is to provide a free open library of 3d models. This requires certain restrictions regarding contributions with respect to their legal status.

- Users must only submit models for which they own the rights allowing relicensing under the KiCad library licence.
- In most cases this means they have created the model themselves, either manually designed in a parametric modeling software or via scripting tools.
- Contributing model data designed by a third party complicates licensing issues and is to be avoided. This includes files downloaded from a manufacturers website.

M1.2 Withdrawn

This rule was withdrawn.

M1.3 Source files for 3D models must be supplied

The KiCad library maintains a repository for 3D source files at https://gitlab.com/kicad/libraries/kicad-packages3D-source.

The *source files* used to generate 3D model data must be submitted to this repository when a user contributes 3D model data.

- Source files of 3d models are the native files generated by the 3d modeling software (Example: fcstd for Freecad).
- Source files alternatively are any generator scripts.
  - However scripts found in https://github.com/easyw/kicad-3d-models-in-freecad do not need to be added to the source repo.
- The source repository directory structure mirrors that of the kicad-packages-3d repository.

M2 - 3D File Requirements

M2.1 3D models must be supplied in both 'step' and 'wrl' formats

Any 3D model contribution to the KiCad library must be made in both step and wrl format.

WRL

- WRL files contain material properties and are used for realistic rendering. However they cannot be exported to MCAD packages.

STEP

- STEP files are used for integration with MCAD software packages. These files do not contain material properties and

cannot be used for realistic rendering.

M2.2 Model alignment and scaling

The 3D models must be aligned and scaled appropriately to match the associated footprint.

Alignment

The 3D model must be aligned such that it does not require an additional alignment offset in the footprint options. When associating a 3D model with a footprint, the `offset` parameter must read (`0`, `0`, `0`)

Scaling

1. STEP files include absolute dimensional information, and *must never have a secondary scaling factor applied*. The model `scaling` parameter in KiCad must read (`1`, `1`, `1`)
2. WRL files do not specify absolute dimensions. The WRL fileformat has no way do specify what its units mean. KiCad considers one WRL unit to be 0.1 inches for historical reasons, so models must be scaled accordingly.
    a. Assuming the model is designed in `mm` then the required scaling factor for WRL export is `1/2.54` (~0.393700787).
    b. The Freecad extension *kicad-stepup* exports WRL files already correctly scaled.

Rotation

The model must be rotated such that no additional rotation is required within KiCad to align the 3D model with the footprint.

M2.3 Freecad is the preferred design tool

The goal of the KiCad 3d model library is to provide a free open library of 3d models. The use of open source design tools is preferred for this very reason.

Freecad is chosen as the preferred tool to be used. Especially in combination with the kicad-stepup extension.
Good alternatives are open scripting options like OpenSCAD or CadQuery. For the latter consider contributing to https://gitlab.com/kicad/libraries/kicad-packages3D-generator
Files created in closed source tools are permitted. If a certain model is available designed in both an open and closed tool then the one from the open tool is preferred and will replace the other one.

KLC Revision History

Revision information for the KLC can be found here.

Next - Revision History

Edit on GitLab