

Padstacks and Allegro SKILL

Product Version 17.2
May, 2018

Copyright Statement

© 2018 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence and the Cadence logo are registered trademarks of Cadence Design Systems, Inc. All others are the property of their respective holders.

Contents

Purpose	4
Audience	4
Overview	4
Understanding Padstacks	4
Allegro SKILL and Padstack	6
Make Padstack Using Allegro SKILL	10
Make SMD Pad Using Allegro SKILL	11
Make Through Pad with a Drill	11
Make a Pad with a Shape Symbol	12
Make a Pad with Flash	12
Make a Blind Buried Via	12
Create SMD with SolderMask	13
Create SMD with Coverlay Masks	13
Create a Donut Pad	14
Create a Chamfer Pad	14
Create Keepout on TOP Layer and Adjacent Layer	15
Find Padstack	16
Get Size of the Padstack	16
Modify a Padstack or Copy a Padstack	17
Summary	19
References	19
Support	19
Feedback	19

Purpose

This application note describes a padstack and the process to read it using Allegro SKILL. It also explains how to make different types of padstacks, such as Through-hole padstack, SMD padstack with Soldermask, and so on.

Audience

This document is intended for Allegro SKILL developers who want to work with padstacks using Allegro SKILL.

Overview

Sometimes, you might have to copy or resize a padstack programmatically using Allegro SKILL. To do this, you need to first understand the concept of a padstack and the procedure to read a padstack using Allegro SKILL. You also need to understand how to create and copy different types of padstacks.

Understanding Padstacks

A padstack is a collection of information that is associated with a pin or a via, used to model different types of pads. You can create new library padstacks with Padstack Designer, which you open using the `padstack_editor` command. Padstack Designer lets you create and edit padstacks and save them to your design, to a library, or to both at once.

A padstack is a file that contains the following information for each layer:

- Pad size and shape
- Drill size and drill display figure
- Mask layers such as Soldermask, Pastemask, and Filmmask

PadPath (**Setup > User Preferences > Paths > Library**) is the location where you store the padstacks (`filename.pad`). Make sure that any pins/vias you want to use are stored in this location. The Modify Padstack list will only show the pads that are used in the design. You can launch Padstack Editor as a stand-alone tool from the Start menu and edit/add the pads you need; then, you store them in the PadPath location. Via list in Constraint Manager should only consist of the via padstacks.

The padstack describes how the pin connects to each layer in the design. Each pin can have any pad type (Regular, Thermal Relief, Antipads, and custom shapes) defined on each ETCH/CONDUCTOR layer of the design. For negative artwork layers, the Layout Editor uses Thermal Reliefs and Antipads. For positive artwork layers, the Layout Editor

uses just the Regular pads. This means that a pin can be put anywhere on the design, and the Layout Editor photoplots the correct pad, whether the location is in an open area or inside a filled shape.

A figure can be a Regular figure, or it can be a Shape Symbol (.ssm) or a For Thermal Relief Flash Symbol (a .fsm file). Shape pads are created as symbols and added to a padstack either upon initial creation or when editing a design.

A symbol in PCB Editor comprises the following two files:

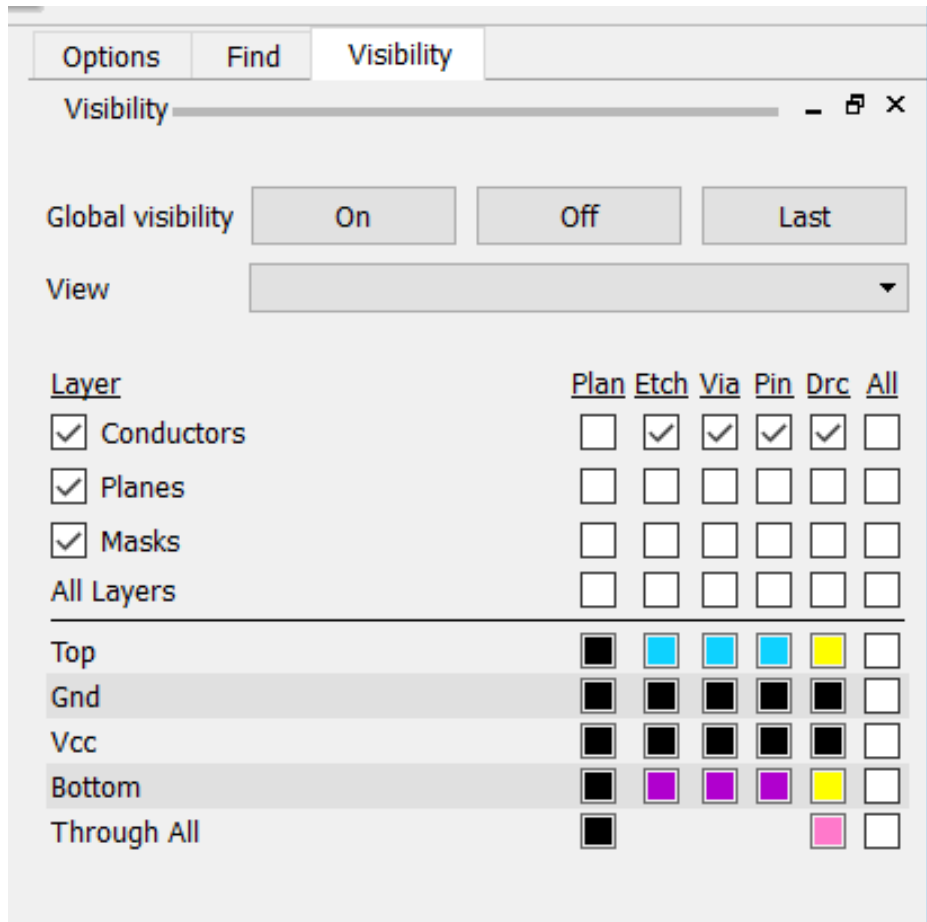
- **The drawing file or dra file:** The drawing file is the result of all the commands that you choose from the Add and Edit selections of the Symbol subset. Additionally, you choose commands from the Layout menu to add a layer of electrical characteristics to the drawing, in the form of connections, pins, labels, and constraints. Drawing file names end with a .dra extension in PCB Editor.
- **The symbol file:** Symbol files result from saving a drawing file and specifying one of the symbol types such as Shape (with .ssm extension) or Flash (with a .fsm extension). To convert the drawing into a symbol, run "create symbol" from an open drawing in the Symbol subset.

To make a padstack using Padstack Editor, you can go through the following videos:

- [Creating a Thru Hole Pad Stack using the Allegro Padstack Editor in 17.2-2016 \(Video\)](#)
- [Creating a Surface Mount Pad Stack using the Allegro Padstack Designer \(Video\)](#)
- [Allegro Front-to-Back User Guide 17.2-2016](#)

Allegro SKILL and Padstack

Let's see what information is available in Allegro SKILL about a padstack and pad. For example, take the padstack for pad `60C85C35D` used in the `cds_routed` board. The board has four layers as shown in the image below:



`axlLoadPadstack` loads a padstack by attempting to find the padstack by name in the existing database. Failing that, Allegro PCB Editor looks in the pad library on the disk.

```
padStackId=axlLoadPadstack("60C85C35D")
```

If it finds the padstack, `padStackId` **will not be NULL**. If you print information about the padstack using `padStackId->??`, you will get the information about drill, layers span, and the list of pads, as shown in the image below:

```
Skill > padStackId=axLoadPadstack("60C85C35D")
dbid:00000253755E47A8
Skill > padStackId->??
(prop nil bbox
  ((-42.5 -42.5)
   (42.5 42.5)
  ) readOnly
  t objType "padstack" startEnd
  ("ETCH/TOP" "ETCH/BOTTOM")
  parent dbid:000002536C2F40C0 name "60C85C35D" plating
  "PLATED" holeType "circle_drill" drillNonStandard nil
  multiDrillData nil backdrillFigureHeight nil backdrillFigureWidth
  nil backdrillFigureChar nil backdrillFigureName nil
  backdrillDiameter nil holeCounterAngle 0 holeCounterDepth
  0.0 holeCounterTolerance
  (0.0 0.0) holeCounterDiameter 0.0
  holeCounterType nil pluralVia nil padSuppression
  t uvia nil usage "Through"
  type "through" keepout nil holeToleranceY
  (0.0 0.0) holeTolerance
  (0.0 0.0) drillFigureHeight 50.0
  drillFigureWidth 50.0 drillChar "" drillFigureName
  "CIRCLE" drillSizeHeight 0.0 drillSizeWidth 0.0
  drillDiameter 35.0 drillOffset
  (0.0 0.0) drillToolSize
  "" isThrough t spanLockCount 0
  derived nil definition nil isPadRef
  nil pads
  (pad:0000025376A90598 pad:0000025376A90570 pad:0000025376A90548 pad:0000025376A90520 pad:0000025376A904F8
   pad:0000025376A904D0 pad:0000025376A904A8 pad:0000025376A90480 pad:0000025376A90458 pad:0000025376A90430
   pad:0000025376A90408 pad:0000025376A903E0 pad:0000025376A903B8 pad:0000025376A90390 pad:0000025376A90368
   pad:0000025376A90340 pad:0000025376A90318 pad:0000025376A902F0
  )
)
```

There are 18 pads in this padstack (the number of elements in the pad list), and information about some of the pads is shown in the image below:

```
Skill > nth(0 pads)->??
(readOnly t objType "pad" figure
 [_axlPath@0x1de065d8068] name nil corners nil
 radius 0.0 sides 0 inside
 0.0 bBox
 [(-30.0 -30.0)
 (30.0 30.0)
 ] parent dbid:000001DE076967A8
 offset
 (0.0 0.0) type "REGULAR" layer
 "ETCH/TOP" flash "" figureName "CIRCLE"
)
Skill > nth(1 pads)->??
(readOnly t objType "pad" figure
 [_axlPath@0x1de065d82a8 _axlPath@0x1de065d8218 _axlPath@0x1de065d8188 _axlPath@0x1de065d80f8] name "TR_80_60" corners nil
 radius 0.0 sides 0 inside
 0.0 bBox
 [(-40.0 -40.0)
 (40.0 40.0)
 ] parent dbid:000001DE076967A8
 offset
 (0.0 0.0) type "THERMAL" layer
 "ETCH/TOP" flash "TR_80_60" figureName "FLASH"
)
Skill > nth(2 pads)->??
(readOnly t objType "pad" figure
 [_axlPath@0x1de065d82f0] name nil corners nil
 radius 0.0 sides 0 inside
 0.0 bBox
 [(-42.5 -42.5)
 (42.5 42.5)
 ] parent dbid:000001DE076967A8
 offset
 (0.0 0.0) type "ANTI" layer
 "ETCH/TOP" flash "" figureName "CIRCLE"
)
}
```

Total Number of pads

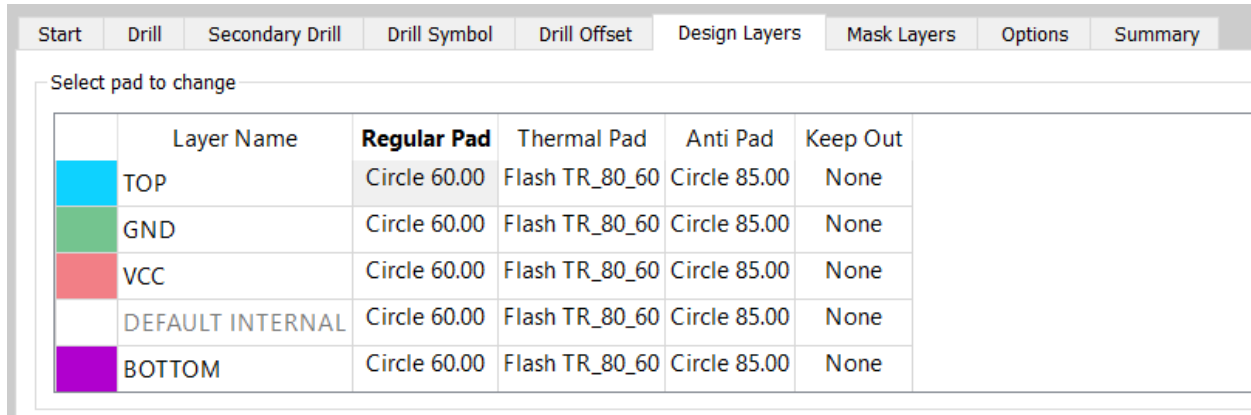
$$= 3 * \text{Number of Layers} + \text{Soldermask_top} + \text{Soldermask_bottom} \\ + \text{pastemask_top} + \text{pastemask_bottom} + \text{filmmask_top} + \\ \text{filmmask_bottom}$$






Number of Layers are 4 (TOP, GND, VCC, and BOTTOM)

$$= 3 * 4 + 6 = 18$$

Padstacks and Allegro SKILL

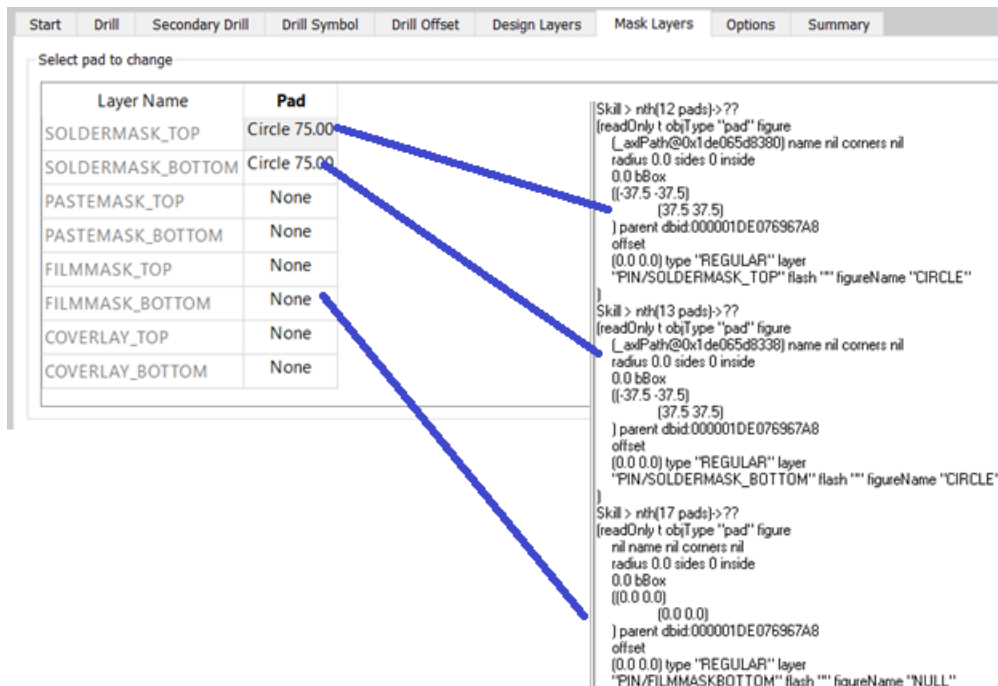
Note that for every layer, three types of pads are defined: Regular, Thermal, and AntiPad. You can confirm it from the Padstack Editor **Design Layers** tab shown in the image below:



	Layer Name	Regular Pad	Thermal Pad	Anti Pad	Keep Out
	TOP	Circle 60.00	Flash TR_80_60	Circle 85.00	None
	GND	Circle 60.00	Flash TR_80_60	Circle 85.00	None
	VCC	Circle 60.00	Flash TR_80_60	Circle 85.00	None
	DEFAULT INTERNAL	Circle 60.00	Flash TR_80_60	Circle 85.00	None
	BOTTOM	Circle 60.00	Flash TR_80_60	Circle 85.00	None

Thermal pad in the above image refers to the Flash symbol TR_80_60. The TR_80_60.fsm file should be present in PSMPATH.

For the maskLayers, the information about the pad from Allegro SKILL and Padstack Editor is shown in the image below:



Layer Name	Pad
SOLDERMASK_TOP	Circle 75.00
SOLDERMASK_BOTTOM	Circle 75.00
PASTEMASK_TOP	None
PASTEMASK_BOTTOM	None
FILMMASK_TOP	None
FILMMASK_BOTTOM	None
COVERLAY_TOP	None
COVERLAY_BOTTOM	None

```
Skill > nth(12 pads)->??
(readOnly t objType "pad" figure
 [_axiPath@0x1de065d8338]) name nil corners nil
radius 0.0 sides 0 inside
0.0 bBox
[(-37.5 -37.5)
 (37.5 37.5)
] parent dbid:000001DE076967A8
offset
(0.0 0.0) type "REGULAR" layer
"PIN/SOLDERMASK_TOP" flash "" figureName "CIRCLE"
}

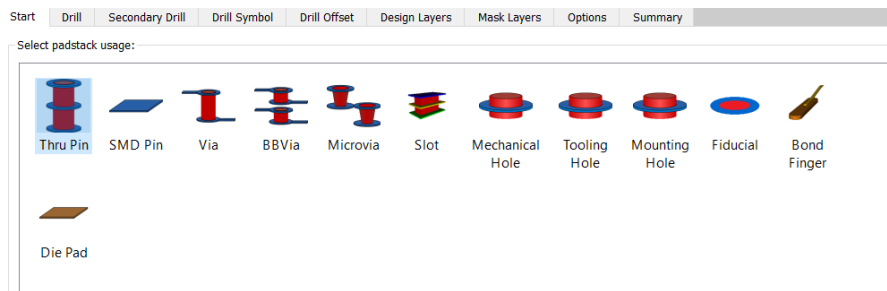
Skill > nth(13 pads)->??
(readOnly t objType "pad" figure
 [_axiPath@0x1de065d8338]) name nil corners nil
radius 0.0 sides 0 inside
0.0 bBox
[(-37.5 -37.5)
 (37.5 37.5)
] parent dbid:000001DE076967A8
offset
(0.0 0.0) type "REGULAR" layer
"PIN/SOLDERMASK_BOTTOM" flash "" figureName "CIRCLE"
}

Skill > nth(17 pads)->??
(readOnly t objType "pad" figure
 nil name nil corners nil
radius 0.0 sides 0 inside
0.0 bBox
[(0.0 0.0)
 (0.0 0.0)
] parent dbid:000001DE076967A8
offset
(0.0 0.0) type "REGULAR" layer
"PIN/FILMMASKBOTTOM" flash "" figureName "NULL"
```

Make Padstack Using Allegro SKILL

Based on the intended use, you can create 12 types of padstacks that are defined in Padstack Editor. Some of the padstacks that can be created are given below:

- **Through:** Refers to a padstack with pads on all ETCH/CONDUCTOR layers. **Select padstack usage** in Padstack Editor has **Thru Pin** selected, as shown in the image below.
- **Blind/Buried Via:** Indicates that the padstack is blind (between the surface and internal layers) or buried (between internal layers). Select **BBVia** in **Select padstack usage**.
- **Microvia:** Refers to the Blind/Buried vias used in high-density interconnect boards and packages. Once placed, design rules examine touch (via tangency) and coincidence (via stacking) of the padstack's pads and other design objects. Select **Microvia** in **Select padstack usage**.
- **Fiducial:** Refers to fiducial markers. Select **Fiducial** in **Select padstack usage**.
- **SMD:** Indicates that the padstack is for SMD pins. Select **SMD Pin** in **Select padstack usage**.



Steps involved in making a padstack are as follows:

1. Make Drill using `make_axlPadStackDrill` (This is required for Through Pad).
2. Make pads using `make_axlPadStackPad` for each pad on each layer. Collect all these pads in a list.
3. Use `axlDBCCreatePadStack` to create a padstack using the list of pads defined in Step 2.
4. Use `axlPadstackToDisk` to write the pad file.

See `<cdns installation>/share/pcb/examples/skill/dbcreate/pad.il` for more examples.

Make SMD Pad Using Allegro SKILL

The following code adds a surface-mount padstack having a 25-by-60 rectangular pad on the top layer:

```
smdpad = make_axlPadStackPad( ?layer "TOP", ?type 'REGULAR,  
?figure 'RECTANGLE, ?figureSize 25:60) nil)  
  
pad_list = cons(smdpad nil)  
  
ps_id = axlDBCreatePadStack("smt_pad", nil, pad_list t)
```

Make Through Pad with a Drill

The following code adds a padstack of Through type:

- With a square drill of 60 and a 42-size plated thru hole:

```
drill_data = make_axlPadStackDrill(?drillDiameter 42  
?figure 'SQUARE, ?figureSize 60:60, ?plating 'PLATED)
```

- 80-diameter circle pad on the top layer:

```
pad_list = cons(make_axlPadStackPad(?layer "TOP", ?type  
'REGULAR, ?figure 'CIRCLE, ?figureSize 80:80) pad_list)
```

- 75-diameter circle pad on internal layers:

```
pad_list = cons(make_axlPadStackPad( ?layer "DEFAULT  
INTERNAL", ?type 'REGULAR, ?figure 'CIRCLE, ?figureSize  
75:75) pad_list)
```

- 80-unit square pad on the bottom layer:

```
pad_list = cons(make_axlPadStackPad( ?layer "BOTTOM", ?type  
'REGULAR, ?figure 'SQUARE, ?figureSize 80:80) pad_list)
```

The following code will create a padstack that ties the drill and the padstack:

```
ps_id = axlDBCreatePadStack("thru_pad", drill_data,  
pad_list t)
```

Use the following code to write the padstack to the disk:

```
axlPadstackToDisk("thru_pad.pad");
```

Make a Pad with a Shape Symbol

Use the name of the shape file (for example, `myshape`) as the name of the figure. The `myshape.ssm` file should be created first before using it in the padstack.

```
pad = make_axlPadStackPad(?layer "TOP",?type `REGULAR,
    ?figure "myshape" ?figureSize 80:80) pad_list)
```

Make a Pad with Flash

Use the name of the flash file (for example, `myflash`) as `?flash` and set the figure to ``FLASH`.

```
pad = make_axlPadStackPad(?layer "TOP", ?type `THERMAL,
    ?figure `FLASH ?flash "myflash" ?figureSize 80:80 )
pad_list)
```

Make a Blind Buried Via

Define a drill and make pads for the layer. In the Blind Via example below, pads are defined for layers `TOP` and `Layer1`.

```
name = "bbvia"

drill_data = make_axlPadStackDrill(
    ?usage "bbvia"
    ?drillDiameter 38
    ?plating `PLATED
    ?drillToolSize "Keith"
    ?spanLockCount t
    ?holeTolerance 1:2
    ?drillNonStandard `LASER_DRILL
)

pad_list = nil
```

```
        pad_list = cons(make_axlPadStackPad(?layer
"TOP",?type 'REGULAR, ?figure 'CIRCLE, ?figureSize 80:80)
pad_list)

        pad_list = cons(make_axlPadStackPad(?layer "DEFAULT
INTERNAL", ?type 'REGULAR,?figure 'CIRCLE, ?figureSize
75:75) pad_list)

        pad_list = cons(make_axlPadStackPad(?layer
"LAYER_1", ?type 'REGULAR,?figure 'SQUARE, ?figureSize
80:80) pad_list)

        ps_id = axlDBCCreatePadStack(name, drill_data,
pad_list t)
```

Create SMD with SolderMask

To create an SMD padstack with the name `solder` which has `solderMask` on the `Top` layer, define a padstack for the `TOP` layer and for the layer `SOLDERMASK_TOP`.

```
        pad_list = nil

        pad_list = cons(make_axlPadStackPad(?layer "TOP", ?type
'REGULAR,?figure 'RECTANGLE, ?figureSize 25:60) pad_list)

        pad_list = cons(make_axlPadStackPad(?layer
"SOLDERMASK_TOP", ?type 'REGULAR,?figure 'RECTANGLE,
?figureSize 35:70) pad_list)

name = "solder"    ps_id = axlDBCCreatePadStack(name, nil,
pad_list t)
```

Create SMD with Coverlay Masks

To create an SMD padstack with the name `coverlay` which has `Coverlay` on the `Top` layer, define a padstack for the `TOP` layer and for `COVERLAY_TOP`.

```
name = "coverlay"

        pad_list = nil

        pad_list = cons(make_axlPadStackPad(?layer "TOP", ?type
'REGULAR,?figure 'RECTANGLE, ?figureSize 25:60) pad_list)
```

```
pad_list = cons(make_axlPadStackPad(?layer
"COVERLAY_TOP", ?type 'REGULAR,?figure 'RECTANGLE,
?figureSize 35:70) pad_list)

pad_list = cons(make_axlPadStackPad(?layer
"COVERLAY_BOTTOM", ?type 'REGULAR,?figure 'SQUARE,
?figureSize 50:50) pad_list)

ps_id = axlDBCreatePadStack(name, nil, pad_list t)
```

Create a Donut Pad

To create a donut pad, define the figure as `DONUT` and specify the inside diameter.

```
name = "donut-1"

insideDia = 20.1

pad_list = cons(make_axlPadStackPad(?layer "TOP" ?type
'REGULAR ?figure 'DONUT ?figureSize 60:60?inside insideDia
) nil)

ps_id = axlDBCreatePadStack(name nil pad_list t)
```

Create a Chamfer Pad

Create a Chamfer pad with `?figure 'CHAMFERED_RECTANGLE`, specify the corner as `"UL-LR"`, and specify the radius.

```
type = 'CHAMFERED_RECTANGLE

name = "chamfer-1"

radius = 20

corners = "UL-LR"

pad_list = cons(make_axlPadStackPad(
                                ?layer "TOP"
                                ?type 'REGULAR
                                ?figure type
```

```
        ?figureSize 60:120
        ?corners corners
        ?radius radius
    ) nil)

ps_id = axlDBCreatePadStack(name nil pad_list t)

printf("PAD dbid %L\n" ps_id)
```

Create Keepout on TOP Layer and Adjacent Layer

Create Keepout by specifying ?type 'KEEPOUT for layer TOP and ?layer 'adjacent.

```
    ; regular

    pad_list = cons( make_axlPadStackPad(?layer "TOP" ?type
'REGULAR ?figure 'CIRCLE?figureSize 60:60) nil)

    ; ko same layer

    pad_list = cons( make_axlPadStackPad( ?layer "TOP" ?type
'KEEPOUT ?figure 'CIRCLE ?figureSize 80:80) pad_list)

    ; ko adjacent layer

    pad_list = cons( make_axlPadStackPad(?layer 'adjacent ?type
'KEEPOUT ?figure 'SQUARE ?figureSize 70:70) pad_list)

    pad_list = cons( make_axlPadStackPad(?layer "GND" ?type
'REGULAR ?figure 'CIRCLE ?figureSize 60:60) pad_list)

pad_list = cons( make_axlPadStackPad( ?layer "GND" ?type
'KEEPOUT ?figure 'CIRCLE ?figureSize 80:80) _list)

name = "ko"

ps_id = axlDBCreatePadStack(name nil pad_list t)
```

Find Padstack

Use `axlDBFindByName` to get the `dbid` of the padstack name. You can use `axlDBFindByName` to find an object by name, without involving the selection set for objects like `net`, `refdes`, or `padstack`. For example, use the code below to find the padstack by the name `thru`:

```
db = axlDBFindByName('padstack "thru")
```

Use `axlDBGetPad (o_dbid t_layer t_type)` to get the pad of type `t_type` associated with layer `t_layer`.

Note: SMD pads will not have a default internal layer.

`t_layer`: String representation of a layer of the pad to retrieve; for example, "ETCH/TOP".

`t_type`: Type of pad to retrieve: "REGULAR", "ANTI", or "THERMAL".

Get Size of the Padstack

To find the size of the pad, you can use the Bounding Box of the pad and find its width and height.

```
(procedure getPadFigureSize(inPad)
  (let (firstCo x1 y1 x2 y2 secCo width height bbox)
    bbox=inPad->bBox
    firstCo=car(bbox)
    x1=car(firstCo)
    y1=cadr(firstCo)
    secCo=cadr(bbox)
    x2=car(secCo)
    y2=cadr(secCo)
    width=abs(x2-x1)
    height=abs(y2-y1)
```



```
        width:height
    )
);procedure
```

Modify a Padstack or Copy a Padstack

Programmatically, you cannot modify a padstack, as it is read-only. So, you need to copy the pads and then make modifications. The function to copy a pad is shown below.

You need to find the layer of the pad and decide the parameters to copy depending on whether the `figureName` is `FLASH` or not. For `FLASH`, copy the name to `?flash`; else, copy `inPad ->figureName` to `figure`.

```
(procedure copyPad(inPad)

  (let (layer subClassName className)

    layer = parseString(inPad ->layer "/" )

    ;printf("layer is %L\n" layer)

    className = car(layer)

    subClassName = cadr(layer)

    size=getPadFigureSize(inPad)

    (if inPad->figureName == "FLASH" then

      newPad = make_axlPadStackPad(?layer subClassName,

        ?type inPad ->type

        ?figureSize size

        ?figure 'FLASH

        ?flash inPad ->flash

        ?offset inPad ->offset

      )

    )

  )
```

```
        else
            newPad = make_axlPadStackPad(?layer
subClassName,
            ?type inPad ->type,
            ?figureSize size
            ?figure inPad ->figureName
            ?offset inPad ->offset
            )
        );if
        newPad
    );let
);procedure
```

Summary

In this application note, you learned about padstacks and pads, how to read them, and how to make them using Allegro SKILL.

References

Allegro SKILL User Guide

Examples at `<cdns
installation>/share/pcb/examples/skill/dbcreate/pad.il`

Support

Cadence Support Portal provides access to support resources, including an extensive knowledge base, access to software updates for Cadence products, and the ability to interact with Cadence Customer Support. Visit <https://support.cadence.com>.

Feedback

Email comments, questions, and suggestions to content_feedback@cadence.com.