# KMK FW - Layers

Layers module adds keys for accessing other layers. It can simply be added to the extensions list.

```
from kmk.modules.layers import Layers
keyboard.modules.append(Layers())
```

## Keycodes

| Key | Description |
|---|---|
| `KC.FD(layer)` | Replaces the top layer |
| `KC.DF(layer)` | Switches the default layer until the next time the keyboard powers off |
| `KC.MO(layer)` | Momentarily activates layer, switches off when you let go |
| `KC.LM(layer, mod)` | As `MO(layer)` but with `mod` active |
| `KC.LT(layer, kc)` | Momentarily activates layer if held, sends kc if tapped |
| `KC.TG(layer)` | Toggles the layer (enables it if not active, and vice versa) |
| `KC.TO(layer)` | Activates layer and deactivates all other layers |
| `KC.TT(layer)` | Momentarily activates layer if held, toggles it if tapped repeatedly |

## Custom HoldTap Behavior

`KC.TT` and `KC.LT` use the same heuristic to determine taps and holds as HoldTap. Check out the HoldTap doc to find out more.

## Working with Layers

When starting out, care should be taken when working with layers, since it's possible to lock yourself to a layer with no way of returning to the base layer short of unplugging your keyboard. This is especially easy to do when using the `KC.TO()` keycode, which deactivates all other layers in the stack.

Some helpful guidelines to keep in mind as you design your layers: - Only reference higher-numbered layers from a given layer - Leave keys as `KC.TRNS` in higher layers when they would overlap with a layer-switch

## Using Combo Layers

Combo Layers allow you to activate a corresponding layer based on the activation of 2 or more other layers. The advantage of using Combo layers is that when you release one of the layer keys, it stays on whatever

layer is still being held. See combo layers documentation for more information on it's function and to see examples.

### Using Multiple Base Layers

In some cases, you may want to have more than one base layer (for instance you want to use both QWERTY and Dvorak layouts, or you have a custom gamepad that can switch between different games). In this case, best practice is to have these layers be the lowest, i.e. defined first in your keymap. These layers are mutually-exclusive, so treat changing default layers with `KC.DF()` the same way that you would treat using `KC.TO()`

## Example Code

For our example, let's take a simple 3x3 macropad with two layers as follows:

```
from kmk.modules.layers import Layers
keyboard.modules.append(Layers())

# Layer Keys
MOMENTARY = KC.MO(1)
MOD_LAYER = KC.LM(1, KC.RCTL)
LAYER_TAP = KC.LT(1, KC.END, prefer_hold=True, tap_interrupted=False,
tap_time=250) # any tap longer than 250ms will be interpreted as a hold

keyboard.keymap = [
    # Base layer
    [
        KC.NO,  KC.UP,  KC.NO,
        KC.LEFT,KC.DOWN,KC.RGHT,
        MOMENTARY, LAYER_TAP, MOD_LAYER,
    ],

    # Function Layer
    [
        KC.F1,  KC.F2,  KC.F3,
        KC.F4,  KC.F5,  KC.F6,
        KC.TRNS,KC.TRNS,KC.TRNS,
    ],
]
```

## Active Layer indication with RGB

A common question is: "How do I change RGB background based on my active layer?" Here is *one* (simple) way of many to go about it.

To indicate active layer you can use RGB background, or in many cases board's status LED, then no additional hardware is needed. Information about the LED type and to which GPIO pin it is connected is often available on the pinout of the board and in the documentation.

In this example on board RGB status LED is used. Number of layers is unlimited and only chosen layers can be used. Note, that LED's basic colors can have different order for different hardware.

```
import board

from kmk.modules.layers import Layers as _Layers
from kmk.extensions.rgb import RGB

sat = 255
val = 5; # brightness in range 0-255

rgb = RGB(pixel_pin=board.GP16,     # GPIO pin of the status LED, or
background RGB light
        num_pixels=1,               # one if status LED, more if background
RGB light
        rgb_order=(0, 1, 2),        # RGB order may differ depending on the
hardware
        hue_default=0,              # in range 0-255: 0/255-red, 85-green,
170-blue
        sat_default=sat,
        val_default=val,
        )

keyboard.extensions.append(rgb)

class Layers(_Layers):
    last_top_layer = 0

    def after_hid_send(self, keyboard):
        if keyboard.active_layers[0] != self.last_top_layer:
            self.last_top_layer = keyboard.active_layers[0]
            if self.last_top_layer == 0:  # default
                rgb.set_hsv_fill(0, sat, val)   # red
            elif self.last_top_layer == 1:
                rgb.set_hsv_fill(170, sat, val) # blue
```

```
            elif self.last_top_layer == 2:
                rgb.set_hsv_fill(43, sat, val)  # yellow
            elif self.last_top_layer == 4:
                rgb.set_hsv_fill(0, 0, val)     # white

keyboard.modules.append(Layers())
```

Made with Material for MkDocs