moergo

Glove80 Layout Editor User Guide

Date:25 September 2023

Table of Contents

Introduction

All of us are different. We have different hand shapes and sizes. We use different applications and we have different habits. We are simply different. Glove80 is designed to adapt to you and your needs, with its highly configurable hardware design and software/firmware as well.

Glove80 Layout Editor is an web application designed to simplify the customization of your Glove80 ergonomic keyboard. With Glove80 Layout Editor you can change the function of each key. It is a complementary service provided by MoErgo, and is subject to its own terms and conditions.

Glove80 Layout Editor is designed to be social, so that the Glove80 user community can share layout and layout ideas. With Glove80 Layout Editor, you can clone a layout shared by another community member, and quickly modify it to make it *your own.*

Glove80 Layout Editor is super easy to use. There is no installation needed. Just create an account and get going within minutes.

Technical requirements

- Web browser: Chrome, Safari or Firefox
- Device: desktop or laptop

Due to the exceptionally broad range of mobile devices, mobile devices are not officially supported. However we aim to provide a decent level of support, and we do test Glove80 Layout Editor with Chrome browser on Android devices.

Alternative methods to generate customized Glove80 firmware

If you would prefer not to use a graphical user interface to create your Glove80 layout, the traditional ZMK methods are there for you.

Please see:

- https://github.com/moergo-sc/glove80-zmk-config to use the Github action to generate ZMK firmware in the traditional ZMK method
- https://zmk.dev/docs/user-setup ZMK documentation on how to generate ZMK firmware, using the standard ZMK method. With Glove80 Layout Editor, you can directly generate the firmware without a Github account, or installing a compiler toolchain.

Getting started

my.glove80.com

In your favorite browser that is supported by Glove80 Layout Editor, visit https://my.glove80.com

That's it. No application installation is required.

Creating an account

You don't need an account to use Glove80 Layout Editor to design a layout and to generate a ZMK keymap file.

However an account is needed to:

- Save your layouts
- Build firmware within the Glove80 Layout Editor

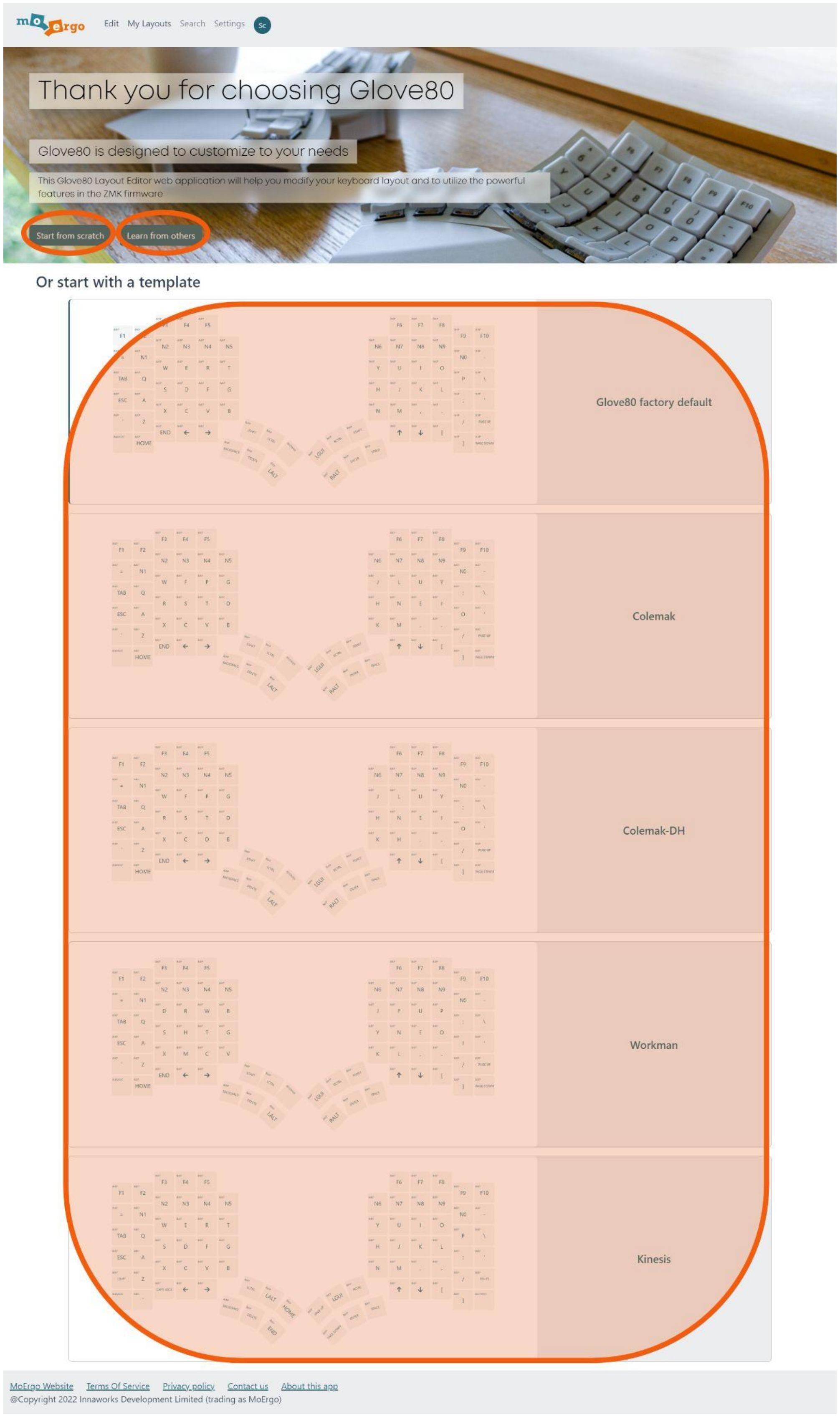
To create a Glove80 Layout Editor account, click on the Sign-Up on the menu bar, or visit https://my.glove80.com/#/signup.

Starting your first Glove80 Layout

From the front page of the Glove80 Layout Editor, you can

- Immediately start editing by clicking on "Start from scratch" button
- Get some inspiration first from the community to see what layouts they have made.
- Use one of the standard templates to get started:
    - Glove80 factory standard layout
    - Glove80 factory standard layout for macOS
    - Colemak
    - Colemak-DH
    - Dvorak
    - Workman
    - Kinesis

If you were previously using a Kinesis Advantage keyboard (just like all of us who designed and tested Glove80), we have created a layout that mimics Kinesis Advantage's default layout to help you migrate. Over time, you may want to take advantage of Glove80's improved thumb cluster by moving more keys into the thumb cluster, but there is no need to rush; you can do it at your own pace.



Thank you for choosing Glove80

Glove80 is designed to customize to your needs

This Glove80 Layout Editor web application will help you modify your keyboard layout and to utilize the powerful features in the ZMK firmware

Or start with a template

Glove80 factory default

Colemak

Colemak-DH

Workman

Kinesis

MoErgo Website Terms Of Service Privacy policy Contact us About this app

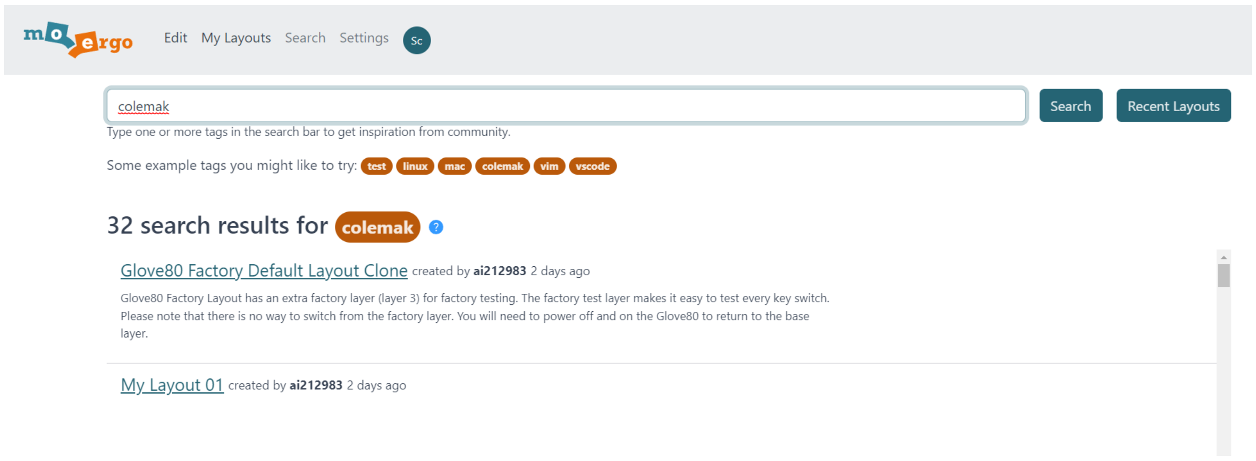
@Copyright 2022 Innaworks Development Limited (trading as MoErgo)

Searching for layouts

Glove80 Layout Editor provides a convenient place for all of us to share our Glove80 layouts.

We found we often get inspiration by looking at layouts made by other Glove80 users. Sometimes other users have already made a layout that already satisfies your needs.

https://my.glove80.com/#/search/



colemak

Type one or more tags in the search bar to get inspiration from community.

Some example tags you might like to try: test linux mac colemak vim vscode

32 search results for colemak

Glove80 Factory Default Layout Clone created by **ai212983** 2 days ago

Glove80 Factory Layout has an extra factory layer (layer 3) for factory testing. The factory test layer makes it easy to test every key switch. Please note that there is no way to switch from the factory layer. You will need to power off and on the Glove80 to return to the base layer.

My Layout 01 created by **ai212983** 2 days ago

You can type one or multiple search terms, separated by commas, and press on the “Search” button to begin the search. Only layouts with tags that match all of the search terms will show up.

Alternatively, click the “Recent Layouts” button to find all recent layouts.

Please note that a Layout is listed only in the search results or recent layouts after its firmware is successfully built.

My Layouts

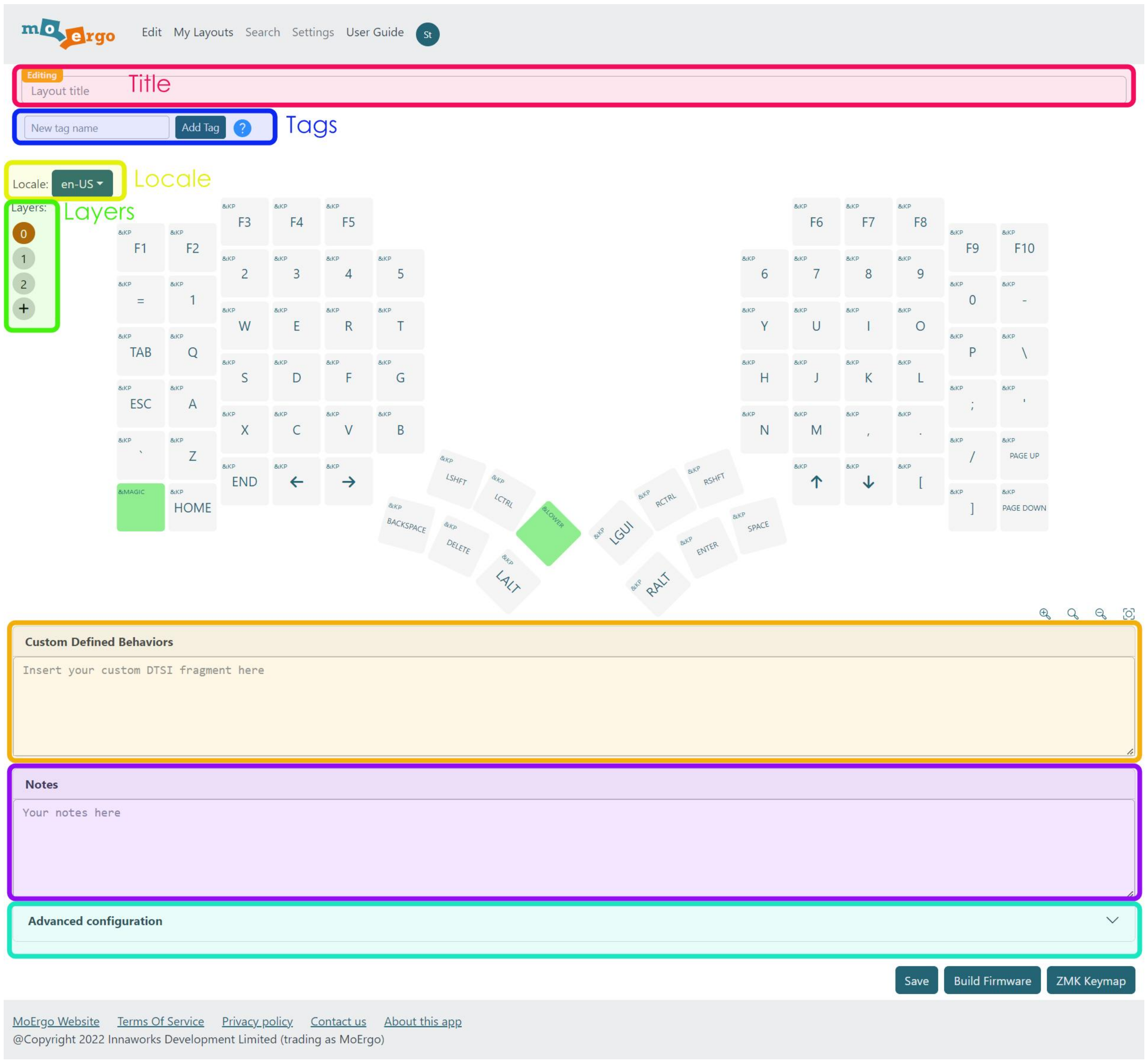
Glove80 Layout Editor keeps a record of your saved layouts. You can retrieve your layouts on any compatible web browser. This functionality is only available if you are logged into your account.

You can reach the My Layouts page by clicking on the “My Layouts” in the menu, or visiting https://my.glove80.com/#/my_layouts/

You can have up to 500 saved layouts.

Layout

When we view a layout, whether it is a template layout, a layout created by another community member or your own saved layout, you will see a screen like below.



Title

A human readable description of the layout.

Tags

Each layout has one or more tags. Tags are descriptive keywords that help yourself remember and to help other people find your layout. A good tag should describe a key distinguishing attribute of the key layout, perhaps describing the application, OS, game that the layout is designed for.

A tag must start with a lower case, and consists of only lower case, numbers and dashes.

Some good examples of tags include:.
OS:
- linux, mac, android

Layout:
- azerty, colemak

Language:
- japanese, german, swedish, norwegian, spanish

Software or games:
- excel, vim, vscode, dota2

Others:
- custom-defined-behaviors

Locale

This changes the rendering of the legends on the keyboard to match the selected locale. This helps international Glove80 users to design their keyboard layouts.

The default locale is en-US, which matches computers set to the United States English locale.

For more information please see International Keyboard Setup, please see International Keyboard Setup.

Layer selector

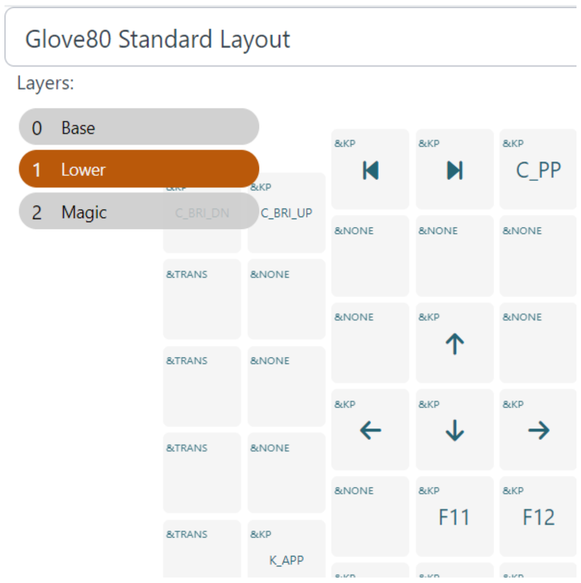
Glove80/ZMK supports the concept of multiple layers. When you power on the keyboard, the base layer (Layer 0) will be active. However, you can switch to different layers; thus, the same key can serve multiple purposes. This isn't as confusing as it may sound, it's similar to the Fn key on a laptop. There are two ways of switching layers: You can switch to another layer momentarily (while you hold down a trigger key, à la "Fn") or indefinitely (until you switch layers again).

When shipped, Glove80's default key layout has 3 Layers:
- Base layer (Layer 0)
- Lower layer (Layer 1)
- Magic layer (Layer 2)

ZMK documentation for layers are at https://zmk.dev/docs/features/keymaps

On Glove80 Layout Editor, the layers are represented on the left hand side. The current layer shown on the layout is the layer highlighted.

When you hover over the circles, the full names of the layers appear and you can select a particular layer to show.



Notes

This optional section is for yourself and others to better understand the intention of the layout.

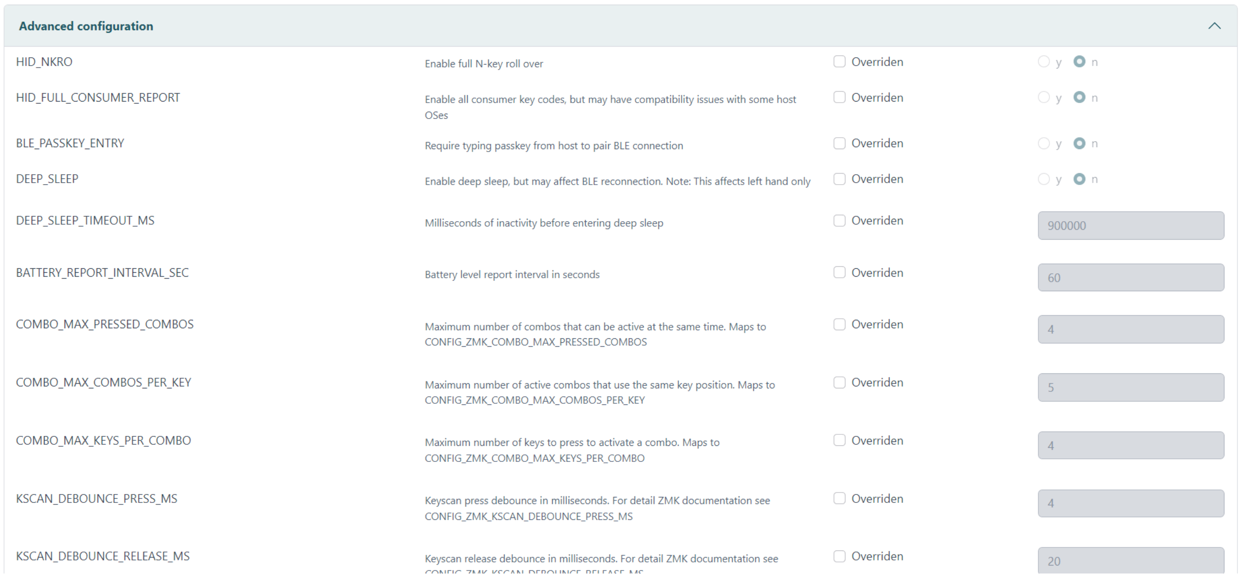
Custom Defined Behavior

This optional section is for injecting your own code into the generated keymap DTSI file. **This is advanced functionality; please use it with extreme caution.**

For more information please see Advanced usage: Custom Defined Behaviors.

Advanced Configuration

The Advanced Configuration section allows you to override the default ZMK firmware behaviors. For most users, it is not necessary to tune these advanced configuration settings.



Advanced configuration

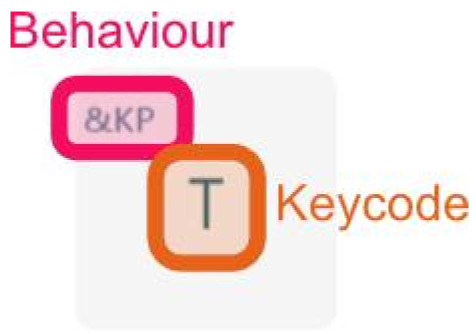
HID_NKRO	Enable full N-key roll over	☐ Overriden	y [✓ n]
HID_FULL_CONSUMER_REPORT	Enable all consumer key codes, but may have compatibility issues with some host OSes	☐ Overriden	y [✓ n]
BLE_PASSKEY_ENTRY	Require typing passkey from host to pair BLE connection	☐ Overriden	y [✓ n]
DEEP_SLEEP	Enable deep sleep, but may affect BLE reconnection. Note: This affects left hand only	☐ Overriden	y [✓ n]
DEEP_SLEEP_TIMEOUT_MS	Milliseconds of inactivity before entering deep sleep	☐ Overriden	900000
BATTERY_REPORT_INTERVAL_SEC	Battery level report interval in seconds	☐ Overriden	60
COMBO_MAX_PRESSED_COMBOS	Maximum number of combos that can be active at the same time. Maps to CONFIG_ZMK_COMBO_MAX_PRESSED_COMBOS	☐ Overriden	4
COMBO_MAX_COMBOS_PER_KEY	Maximum number of active combos that use the same key position. Maps to CONFIG_ZMK_COMBO_MAX_COMBOS_PER_KEY	☐ Overriden	5
COMBO_MAX_KEYS_PER_COMBO	Maximum number of keys to press to activate a combo. Maps to CONFIG_ZMK_COMBO_MAX_KEYS_PER_COMBO	☐ Overriden	4
KSCAN_DEBOUNCE_PRESS_MS	Keyscan press debounce in milliseconds. For detail ZMK documentation see CONFIG_ZMK_KSCAN_DEBOUNCE_PRESS_MS	☐ Overriden	4
KSCAN_DEBOUNCE_RELEASE_MS	Keyscan release debounce in milliseconds. For detail ZMK documentation see	☐ Overriden	20

Some of the Advanced Configuration settings include:

- Enabling or disabling N-key rollover (NKRO)
- Enabling, disabling and configuring deep sleep
- Key debouncing characteristics to eliminiate key chatter
- Debugging and troubleshooting settings

Key binding & key behaviors

Each key in each layer can be bound to a key behavior.



The upper left corner shows the behavior type.

The middle shows the keycode or the behavior parameter, depending on the type of behavior.

Behavior types

Glove80 Layout Editor currently supports the following ZMK behavior types directly.

- `&kp`: key press
- `&bootloader`: enter bootloader mode
- `&bt`: bluetooth
- `&cap_word`: capitalize words
- `&key_repeat`: repeat key
- `&kt`: key toggle
- `&lt`: layer tap
- `&mo`: momentary layer
- `&mt`: mod tap
- `&none`: do nothing
- `&output`: output selection
- `&reset`: reset keyboard
- `&rgb_ub`: RGB underglow control
- `&sk:` sticky key
- `&sl`: sticky layer
- `&to`: to layer
- `&trans`: transparent

For more information on the ZMK key behaviors, please see ZMK documentation at https://zmk.dev/docs

In additional to the above standard ZMK behaviors, Glove80 also supports the following pseudo behaviors:

- `&bt_0`: select bluetooth profile 0
- `&bt_1`: select bluetooth profile 1
- `&bt_2`: select bluetooth profile 2
- `&bt_3`: select bluetooth profile 3
- `&magic`: tap to show RGB indicators. Press to momentarily switch to layer 2 (which is magic layer in the factory layout)
- `&lower`: hold to momentarily switch to layer 1 (lower layer in the factory layout). Double tap to persistently switch to layer 1.
- `Custom`: A special pseudo behavior to allow arbitrary key mapping, to be used with Custom Defined Behavior. For more information please see Custom Defined Behavior.

Keypress behavior: modifier functions

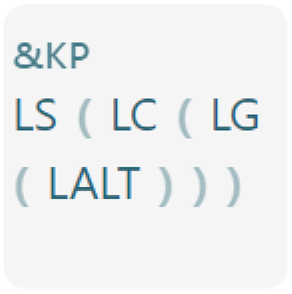
The keypress (`&kp`) behavior in ZMK has a very powerful but not-well-known capability known as the modifier function. Modifier functions make it possible to add one or more modifiers to a keycode.

ZMK documentation for modifier functions are at
https://zmk.dev/docs/codes/modifiers#modifier-functions

Glove80 Layout Editor has built-in support for modifier functions. For example if you type LA as the keycode for an `&kp` behavior, you have selected the Left Alt function. You can then add another key or modifier function.

For example, to define Hyper, you can use `&kp LS(LC(LG(LALT)))`. To do so:

1. Select `&kp` as behavior
2. In the keycode slot, select `LS` as keycode
3. In the keycode slot for `LS` modifier function, select `LC` as keycode
4. In the keycode slot for `LC` modifier function, select `LG` as keycode
5. In the keycode slot for `LG` modifier function, select `LALT` as keycode



`&magic` pseudo behavior

This is a special MoErgo-defined pseudo behavior defined as a macro. For a key bound to this behavior,

- A quick tap will activate the LED indicators. Please see the Glove80 User Guide for more information.
- While holding the key, Glove80 will momentarily switch to layer 2 (the magic layer on a standard layout). After key release, Glove80 will automatically revert back to layer 0

On a standard Glove80 layout, LH C6R6 key is mapped to the `&magic` behavior in all layers.

`&lower` pseudo behavior

This is a special MoErgo-defined pseudo behavior defined as a macro. For a key bound to this behavior,

- While holding the key, Glove80 will momentarily switch to layer 1 (the lower layer on a standard layout). After key release, Glove80 will automatically revert back to layer 0
- If double tapped, Glove80 will permanently switch to layer 1 (the lower layer on a standard layout)

On a standard Glove80 layout, LH T3 key is mapped to the `&lower` behavior in the Base Layer (layer 0).

Bluetooth `&bt_0 &bt_1 &bt_2 &bt_3` pseudo behaviors

These 4 pseudo behaviors switch Glove80 to the respective designated bluetooth device, if the bluetooth device is currently connected.

Underneath the bonnet, these behaviors are implemented as a macro that sets output to bluetooth, and then sets the bluetooth device to the designated bluetooth device.

As an example `&bt_0` is defined as:

```
macros {
    bt_0: bt_profile_macro_0 {
        label = "BT_0";
        compatible = "zmk,behavior-macro";
        #binding-cells = <0>;
        bindings
            = <&out OUT_BLE>,
              <&bt BT_SEL 0>;
    };
};
```

Bluetooth clear all profiles `&bt BT_CLR_ALL`

The `&bt` behavior has been extended with an additional parameter `BT_CLR_ALL`

When bound to `&bt BT_CLR_ALL`, the key will initiate clearing all Bluetooth bindings (excluding the binding to the Glove80 right hand).

Sharing layouts

To share a saved layout simply copy the URL to forward to the person to share with.

Printing layouts

When you are learning a new layout, it is often useful to print a copy of the layout on paper. Glove80 Layout Editor supports this functionality.

You can print a layer by:

1. Selecting the layer you want to print
2. Using the browser print function to print the layer.
   On a Windows browser this is typically activated by `Ctrl-P`

International Keyboard Setup

If you are a user typing in a language other than US English, this section is for you.

A background on Keyboard, HID and Locale

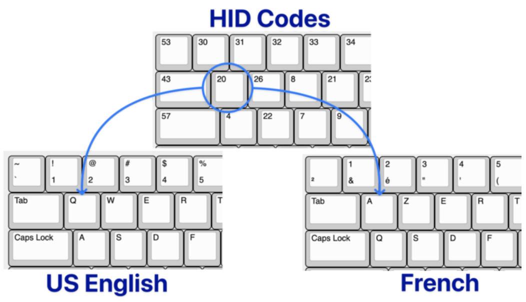
Imagine a standard keyboard. On that keyboard, there are keys with letters, numbers, and symbols. These keys don't have actual letters printed on them; they are more like buttons with no words. To your computer, this is actually how all keyboards appear: all that the computer ‘sees’ when you type is the positions of the keys you press.

When you press one of these keys, like the key marked with the letter 'A', you'd expect the letter 'A' to appear on your screen. But here's the catch: keyboards in different countries have different letters and symbols on the same keys.

So, to make sure your computer understands what you're trying to type, there's a unique code for each key position on the keyboard[1]. When you press a key, the keyboard sends this unique code to the computer. This unique code is known as the **HID keycode**.

Now, your computer knows these unique codes for all the key positions on all sorts of keyboards. But it also needs to know which language or country you're using, because different countries have different letters and symbols on the same keyboard keys. (Some international keyboards also add extra keys with additional unique codes that are not present on all keyboards. For example, the German keyboard has a key marked “<>” whose code is not used on US English keyboards.)

So, your computer has something like a translation book that tells it what each code means in different languages or countries. Let's say you're using a French keyboard, and you press the key in the top left that’s marked 'A.' Your keyboard sends the unique code for that position to the computer. The translation book says, "When you get this code from a French keyboard, show the letter 'A'. But on a US English keyboard, the translation book would instead say to show the letter ‘Q’. The translation book is the **locale.**



[1] Historically, this is because the same internal electronics that sent fixed keycodes were used for keyboards in different countries, and only the printed labels on the keycaps differed. With a programmable keyboard like Glove80, any key position can be configured to send any keycode, but the meaning of each keycode it sends is still interpreted by the computer’s locale settings.

That's how your computer knows which letter or symbol to show on the screen, even if you have a keyboard from another country. The locale set on your computer must match the locale of your keyboard. If the locales of the two ends are not matched, the computer will interpret a HID keycode differently from how you intended.

What locale to set to?

You should set the locale of your layout to match the locale set on your devices.

Currently Glove80 Layout Editor supports these locales:

- en-US: US English keyboard. This is the default locale
- da-DK: Danish keyboard (BETA)
- de-DE: German keyboard (BETA)
- fr-FR-BÉPO: France BÉPO keyboard (BETA)
- jp-JP: Japanese 日本語 keyboard (BETA)
- nb-NO: Norwegian Bokmål keyboard (BETA)
- sv-SE: Sweden keyboard (BETA)

However we expect more locales to be added in the future.

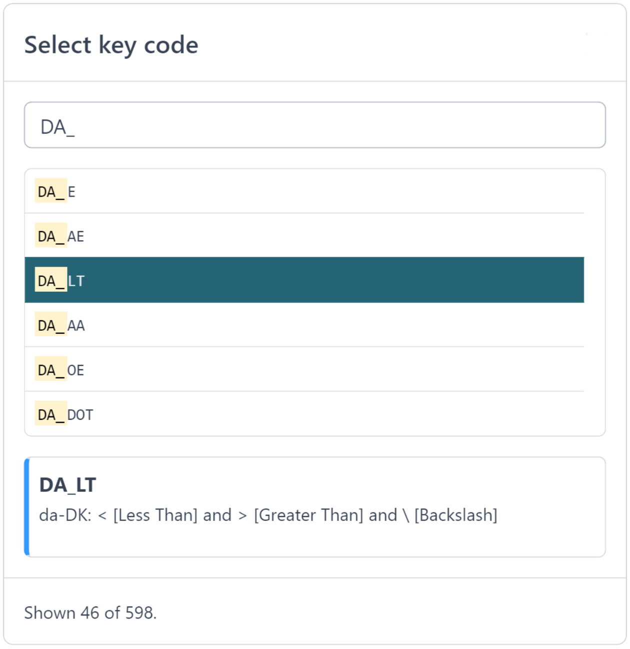
Keycodes

When you select a locale that is not en-US, you will notice the keycodes available for selection would be changed to match the locale.

Some of the en-US keycodes are hidden to avoid confusion. A new set of locale specific keycodes are available for selection These locale specific keycodes are prefixed with the language code:

- da-DK: DA_
- de-DE: DE_
- fr-FR-BÉPO: BÉPO_
- jp-JP: JP_
- nb-NO: NB_
- sv-SE: SV_

For example, for da-DK locale, some of the locale specific keycodes (which are prefixed with DA_) are as follows:



Select key code

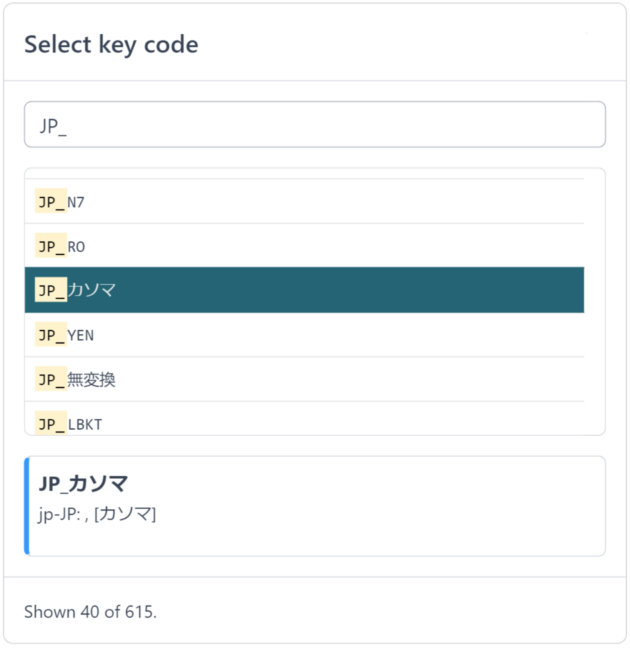
DA_

DA_E
DA_AE
DA_LT
DA_AA
DA_OE
DA_DOT

DA_LT
da-DK: < [Less Than] and > [Greater Than] and \ [Backslash]

Shown 46 of 598.

For example, for ja-JP locale, some of the locale specific keycodes (which are prefixed with JP_) are as follows:



Select key code

JP_

JP_N7
JP_RO
JP_カソマ
JP_YEN
JP_無変換
JP_LBKT

JP_カソマ
jp-JP: , [カソマ]

Shown 40 of 615.

Building firmware

This functionality requires logging into your account.

Glove80 Layout Editor provides a quick and simple way to build firmware. Unlike the traditional ZMK approach, there is no need to create a Github repository, nor to install a compiler toolchain on your computer.

To build firmware for a layout you are viewing, click on the Build Firmware button.

To build firmware for a layout you are currently editing, click on the Build Firmware button. Upon successful generation of firmware, a copy of the layout will be automatically saved and you will be redirected to view the saved layout.

Loading the firmware onto the Glove80

Please see the Glove80 User Guide (https://www.moergo.com/files/glove80-user-guide.pdf) for details.

Limits on number of builds

To ensure that everyone has a fair opportunity to build firmware through the Glove80 Layout Editor, there are daily and weekly limits on the number of builds a user can build.

It is unlikely that you will ever reach these limits. However if you do reach the limit, please wait a while.

Advanced usage: Custom Defined Behaviors

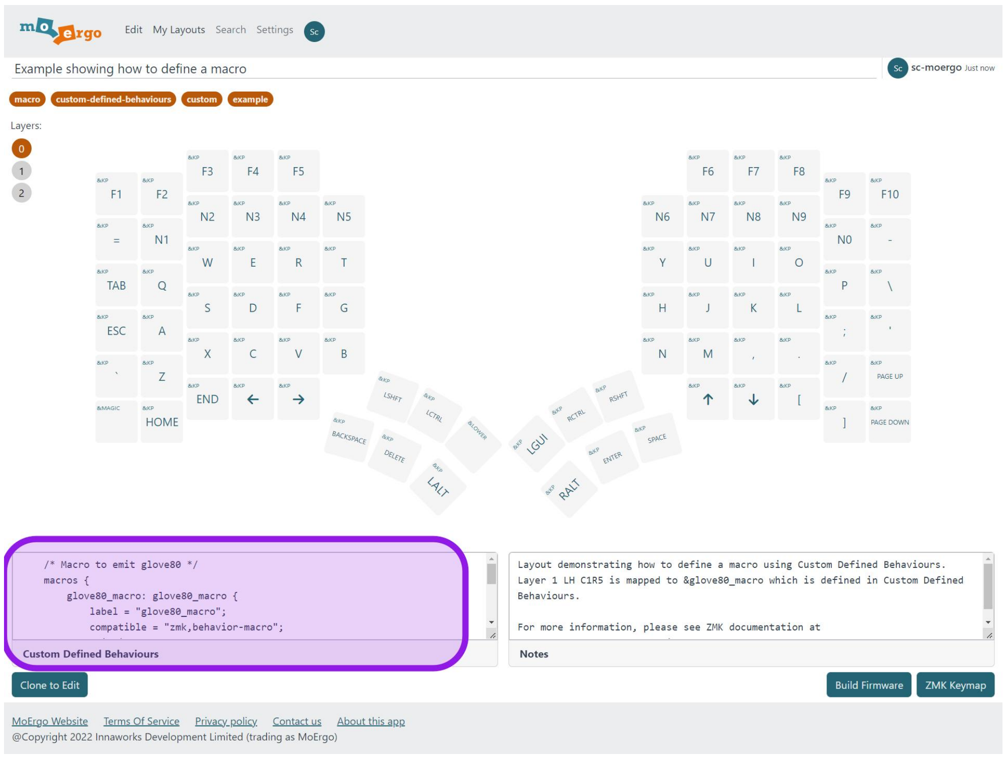
The current version of Glove80 Layout Editor does not support every ZMK feature directly. Missing features include combo and macros.

However these functions can still be supported via Glove80 Layout Editor's Custom Defined Behavior feature.

This is an advanced feature that allows a layout designer to inject text into the keymap DTSI file. It is very powerful and also requires a good understanding of the ZMK DTSI keymap file.

Custom Defined Behaviors

The layout has a section named Custom Defined Behaviors. This is where you define the text to inject into the keymap DTSI file.



Example showing how to define a macro

macro custom-defined-behaviours custom example

```
    /* Macro to emit glove80 */
    macros {
        glove80_macro: glove80_macro {
            label = "glove80_macro";
            compatible = "zmk,behavior-macro";
```

```
Layout demonstrating how to define a macro using Custom Defined Behaviours.
Layer 1 LH C1R5 is mapped to &glove80_macro which is defined in Custom Defined
Behaviours.

For more information, please see ZMK documentation at
```

For example, you could try the following text to define a macro named &glove80_macro. This macro will "type" glove80.

```
/* Macro to emit glove80 */
macros {
    glove80_macro: glove80_macro {
        label = "glove80_macro";
```

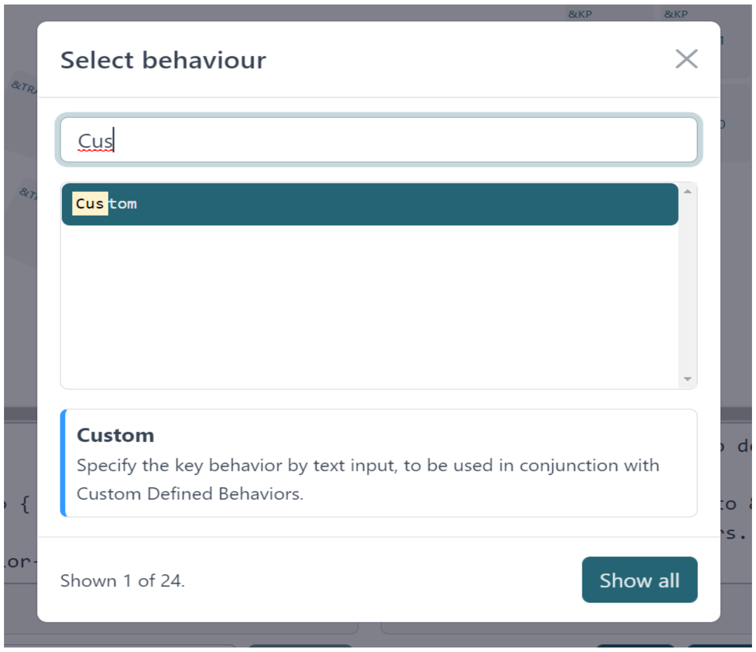
```
            compatible = "zmk,behavior-macro";
            #binding-cells = <0>;
            wait-ms = <40>;
            tap-ms = <40>;
            bindings
            = <&kp G &kp L &kp O &kp V &kp E>
            , <&kp N8 &kp N0>
            ;
        };
    };
```

In general, if your Custom Defined Behaviors refer to a layer, the layer number should not be hardcoded as a value. If layers are re-ordered, the hardcoded layer numbers will be wrong. It is better to use the symbol reference. See Reference to layers for details.

Key binding for Custom Defined Behavior

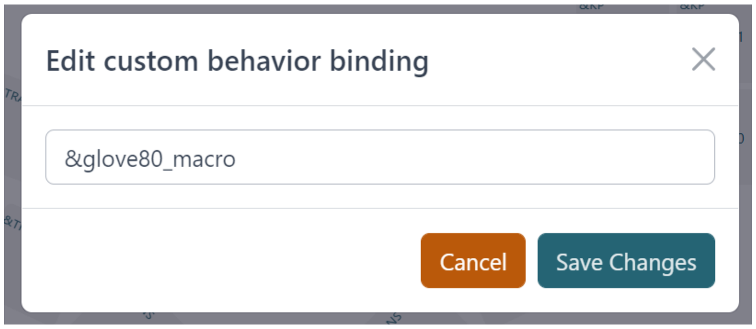
To bind a key to a Custom Defined Behavior you have defined:

1. Click on the top left of the key to select the behavior
2. Choose Custom



Custom
Specify the key behavior by text input, to be used in conjunction with Custom Defined Behaviors.

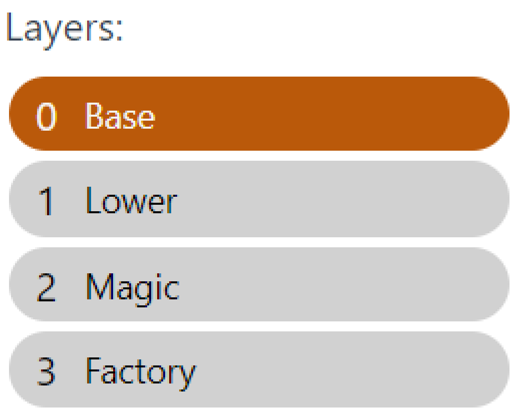
3. Then in the parameter type the actual binding.
   For example, if you want to bind to a macro named `&glove80_macro`, then type the binding as "`&glove80_macro`".



Reference to layers

The exported ZMK keymap automatically generate C #define statement to define each layer. This allows the code in your Custom Defined Behavior to refer to layers symbolically.

For example:
The factory default layout has 4 layers defined



The automatically generated keymap has the following #define:

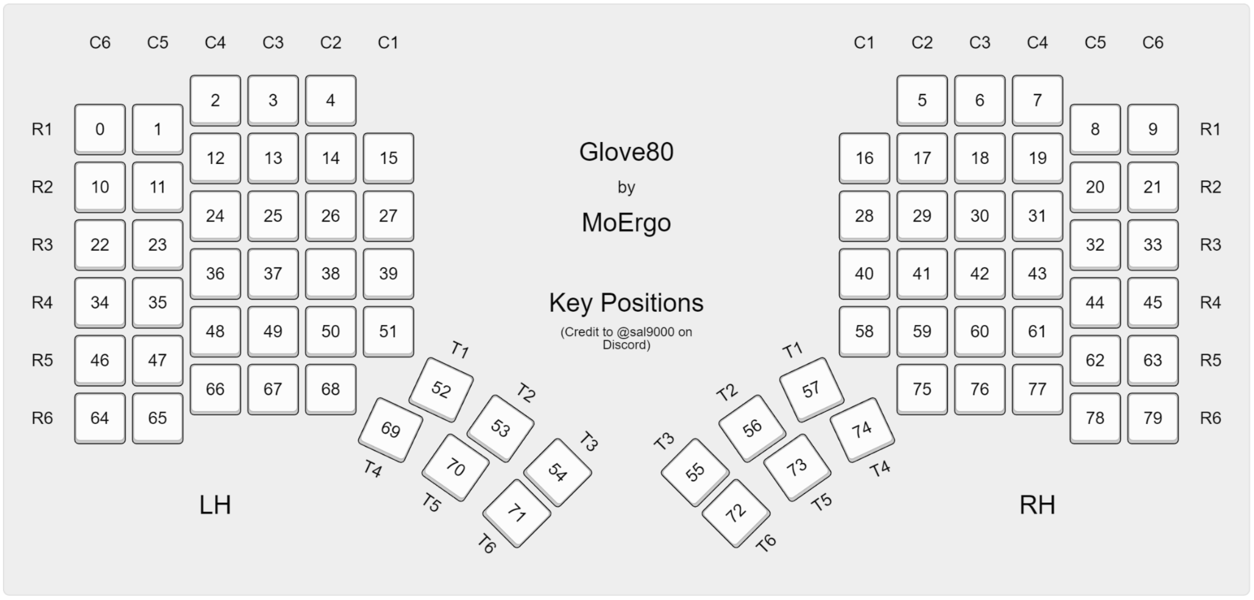
```
/* Automatically generated layer name #define */
#define LAYER_Base 0
#define LAYER_Lower 1
#define LAYER_Magic 2
#define LAYER_Factory 3
```

An example custom behaviour code snippet that uses the generated layer #define

```
/* A modified &lower function that switches to the layer with the name Lower rather
than layer 1 */
    behaviors {
        realLower: realLower {
            compatible = "zmk,behavior-tap-dance";
            label = "LAYER_TAP_DANCE";
            #binding-cells = <0>;
            tapping-term-ms = <200>;
            bindings = <&mo LAYER_Lower>, <&to LAYER_Lower>;
        };
    };
```

Key positions

Certain ZMK features such as Combo (https://zmk.dev/docs/features/combos) and Hold-Tap identify the keys using their key positions.



You can also use the pre-defined #define in the Custom Defined Behaviors to refer to key positions. This is the recommended approach. The macro names are in the format of `POS_<LH|RH>_<position>`, such as:

```
#define POS_LH_T1 52
…
#define POS_LH_T6 71
#define POS_LH_C1R2 15
…
#define POS_LH_C1R5 51
#define POS_LH_C2R1 4
…
#define POS_LH_C2R6 68
…
#define POS_RH_T1 57
…
#define POS_RH_T6 72
#define POS_RH_C1R2 16
…
#define POS_RH_C1R5 58
#define POS_RH_C2R1 5
…
```