# CAPSIFESTO

In Version 15, Keyman extended the ability to use the CAPS key to touch keyboards. This is big news, as CAPS Lock is one of the fundamental features of most mobile keyboards. This document walks through the process of managing caps on your Keyman keyboard.

*Note: This article is written from a roman-script perspective. When I use words like “usually” or “typically”, I am not referring to scripts that do not use upper and lower case forms.*

## Part 1: Shift

### Part 1.1: Shift on Physical Keyboards

The precursor to `caps` is `shift`, so let’s start there. A physical keyboard usually shows a Capital letter (Icon

Description automatically generated) and yet outputs a lowercase letter (`a`) by default.­ If I hold `shift` on my keyboard and press `a` (A picture containing shape

Description automatically generated), I expect the keyboard to output `A`. The user must remember that both `a` and `A` can be accessed via the same key.

In Keyman code, that would be: `+ [K\_A] > a` and `+ [SHIFT K\_A] > A`.

### Part 1.2: Shift on Mobile Keyboards

Mobile keyboards have an advantage that the keyboard can [usually] show you what character will be output. The default layer for most keyboards will be `default`, which means lower case. As expected, pressing the  outputs an `a`.

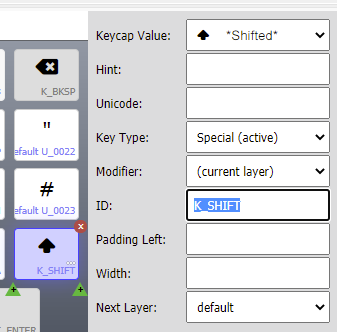


Pressing Shift () on a mobile keyboard [usually] takes the user to a new layer that [mostly] contains upper-cased letters. As expected, pressing  outputs `A`.

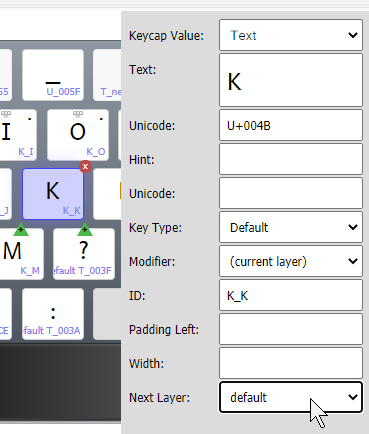


The shift effect on a physical keyboard disappears as soon as one lets go of the shift key. On a mobile keyboard, tapping (not holding) shift activates a new layer and waits for you to press a key. When does shift “end” on a mobile keyboard. For many keyboards (Google’s Gboard, Microsoft’s Swype, iOS), the expectation is that the keyboard will return to the default layer after the user chooses a key or presses the shift key again. This simplifies input for the user because, in normal use of many languages, only the first character of a sentence or proper noun is capitalized.

To show the user that shift is active on the shift layer, make sure you set the Keycap Value to `\*Shifted\*`, The Key Type to `Special (active)`, and the Next Layer to `default`.



In Keyman touch, jumping back to the `default` layer is achieved by choosing a Next Layer from the key properties. After doing whatever the key is programmed to do, the keyboard jumps to the desired layer ready for action! Typically, you’ll want to give all letters on your `shift` layer a Next Layer of `default`.



## Part 2: Caps

If all your capital letters on the `shift` layer return to `default`, how do you write in all capitals?

### Part 2.1: Caps on Desktop

Caps Lock (caps) has been available for desktop keyboards nearly as long as Keyman has been around. Toggling on `caps` allows one to type a string of capital letters until `caps` is turned off again.

This seems simple, but it gets complex quickly. For example, what is the expected output of `[SHIFT CAPS K\_2]`? In versions of Keyman before 14, coding a keyboard requited mental gymnastics as above to make sure that caps only affected certain keys. Defining every possible combination for each key (`NCAPS` means non-caps) looked like this (or worse):

```

+ [NCAPS K\_A] > 'a'  
+ [SHIFT CAPS K\_A] > 'a'  
+ [CAPS K\_A] > 'Α'  
+ [SHIFT NCAPS K\_A] > 'Α'  
+ [NCAPS RALT K\_A] > 'ɛ'  
+ [CAPS SHIFT RALT K\_A] > 'ɛ'  
+ [CAPS RALT K\_A] > 'Ɛ'  
+ [NCAPS SHIFT RALT K\_A] > 'Ɛ'

```

### Part 2.2: Which keys will be affected by the Caps Lock key?

Let’s look at some world keyboards. As a gross oversimplification, much of the English-speaking world uses QWERTY-based keyboards evolved from the US keyboard. Much of the German-speaking world uses QWERTZ-based keyboards. Lastly, much of the French-speaking world uses AZERTY-based keyboards. When developing minority keyboards, you may wish to use common keyboards that the user will already be familiar with to aid transition.

*Note: The real distinction may not be QWERTY/AZERTY, but a dispute between ANSI (QWERTY) and ISO (others) standards for Keyboard layouts.*

Things were greatly simplified when a new feature called `&CasedKeys` was added. In the US QWERTY keyboard (in Microsoft Windows), shift affects every non-modifier key, but the ONLY keys affected by the Caps Lock key are the 26 letter keys, A-Z. If you want to follow QWERTY, the `&CasedKeys` command to be added among the introductory variables would be:

```

store(&CasedKeys) [K\_A]..[K\_Z]

```

In AZERTY and QWERTZ keyboards, Caps affects EVERY non-modifier key except: `K\_BKQUOTE`, `K\_oE2`, `K\_SPACE`, and `K\_NPDOT`. If you want to follow this, the CasedKeys command would be:

```

store(&CasedKeys) [K\_A]..[K\_Z] [K\_0]..[K\_9] [K\_HYPHEN] [K\_EQUAL] [K\_LBRKT] [K\_RBRKT] [K\_BKSLASH] [K\_QUOTE] [K\_COMMA] [K\_PERIOD] [K\_SLASH] [K\_COLON]

```

After defining CasedKeys as above, one can define a simpler and more readable set of keys (see below) and the compiler will automagically decide what to do with caps and caps-shift.

```

+ [K\_A] > 'a'  
+ [SHIFT NCAPS K\_A] > 'Α'  
+ [RALT K\_A] > 'ɛ'  
+ [SHIFT RALT K\_A] > 'Ɛ'

```

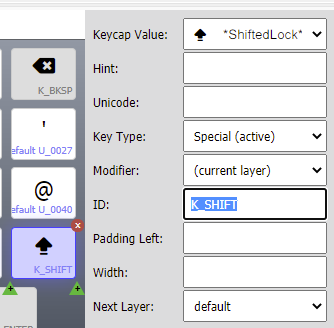
### Part 2.3: Caps on Mobile.

As of Keyman 15, double-tapping on the shift key on `default` or `shift` layer will activate the `caps` layer if there is one. In Keyman 16 (16.0.96 or later), you can jump to an arbitrary layer from the shift keys.

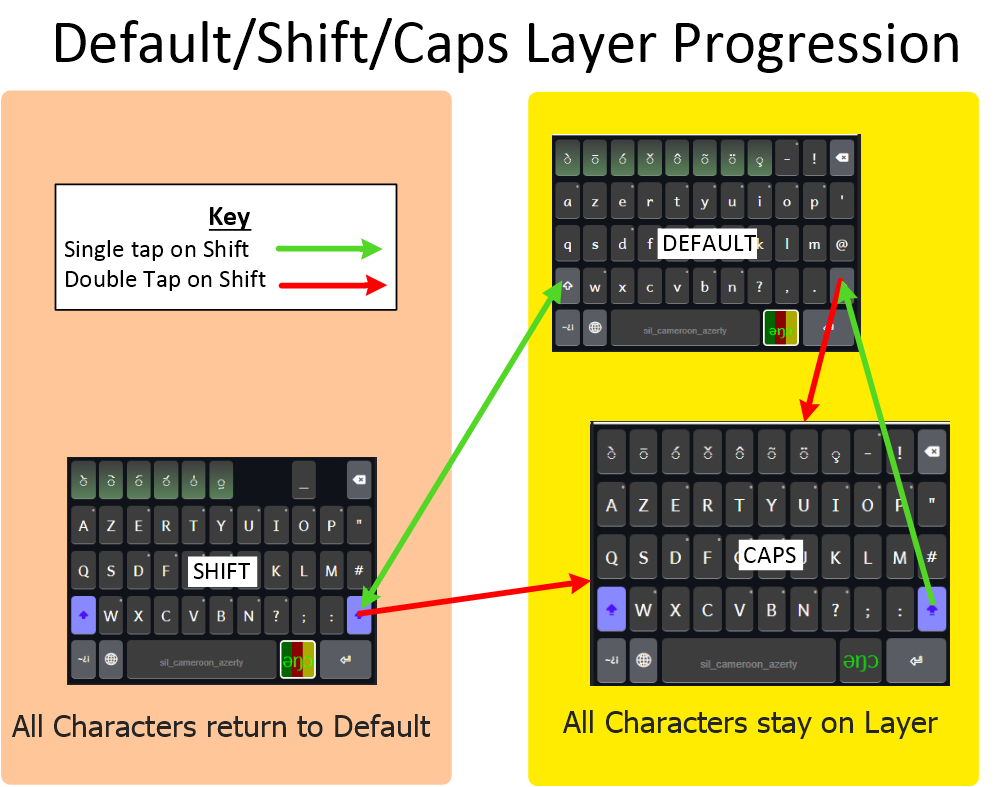
This means that simple or complex mobile keyboards can now take advantage of Caps Lock via double-taps, in a way that will be familiar to most users.

In Keyman Developer, navigate to your shift layer and click on the plus to duplicate that layer. You’ll want to call it `caps`.

Set the shift button(s) on your new `caps` layer to return to `default`, the Keycap Value to `\*ShiftedLock\*`, the Key Type to `Special (active)`, and Next Layer to `default`.



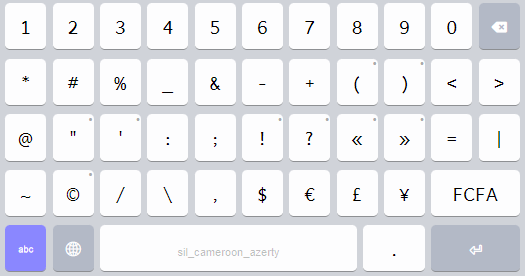
You’ll want to make sure that you’ve removed the Next Layer option from each letter so that the keyboard stays on the same layer. Now that you have a `caps` layer, you can already access it by double-tapping on the `shift` key of the `default` or `shift` layer. If you only have three layers, you’re done!



## Part 3 Symbol Layers

### 3.1 Base Symbol Layer

Maybe you have a symbol layer in your keyboard.



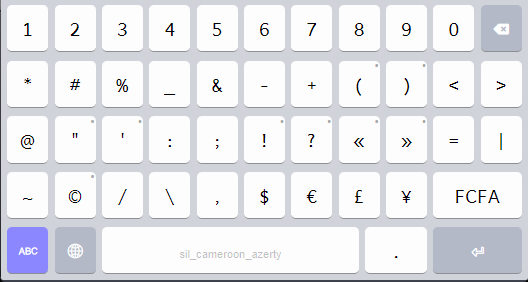
When you come to this layer, you might want to type a number or may only be looking for punctuation. I chose to have most of the punctuation symbols return to default, except for those that are related to numbers. For example, `1,000.57`, `\*69` and `555-1234` can all be typed without leaving the number layer, but `!` or `?` return back to `default`. It’s not perfect, as `:` can either be phrase punctuation or a joiner for hours and minutes (`3:30`)

### 3.2 Symbol Caps

At first thought, you might think that you don’t need a caps version of the symbol layer, but you might. Imagine the Symbol layer is accessible from `default`, `shift`, `and `caps`, where would you expect to return if you go from `caps` to `symbol` and back? With a single symbol layer, you would probably return to `default`, but to me it makes sense to return to `caps`.

For that purpose, I created a symbol-caps layer. The only differences are:

1. The phase punctuation has a next Layer of `caps`.
2. This is a subtle detail; the return button has a capitalized label of `ABC` and returns to ‘caps’ instead of `default`.



## Part 4: Special Character Layers:

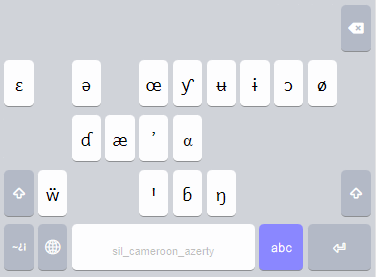
The Cameroon Keyboards are country-wide, with dozens of characters and potentially supporting hundreds of languages.



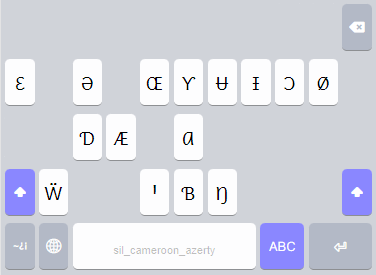
On the Desktop, the letters beyond A-Z are accessed through dead-keys, which works well after the user learns to find the few extra letters used in each language. On the Touch keyboard, each “special” character can be accessed via a long-press on the same key.



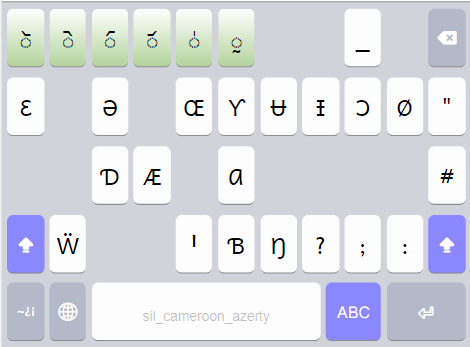
This method makes transition easy for users of the desktop keyboard, but we can do better on touch keyboards. Why not show the possible characters? After pressing the “Cameroon Key” , you will see the `rightalt` layer that presents the lowercase letters not shown on `default`:



From `rightalt`, Each letter has a Next Layer of `default`. `Shift` goes to `rightalt-shift`, and `abc` goes back to default. Double-tapping on shift goes to `rightalt-caps`. Symbol goes to `symbol`.



From `rightalt-shift`, each letter has a Next Layer of `default`. `Shift` goes to `rightalt`, and `ABC` goes back to `shift`. Double-tapping on shift goes to `rightalt-caps`. Symbol goes to `symbol`.



Lastly, from `rightalt-caps`, each letter has a Next Layer of `caps`. Shift cancels caps and goes to `rightalt`, and `ABC` goes to `rightalt`. Double-tapping on shift goes to `rightalt-caps`. Symbol goes to `rightalt-symbol`.

## Part 5: Bonus: Non-Letters and Caps:

The most common combining tones are found on the top row of the `default` layer () and are completed by the ones on the `shift` layer (). If my `caps` layer was exactly like the `shift` layer, it would make typing simple tones more difficult. I chose to put a set of basic tones on the `caps` layer and complex tones on the `rightalt-caps` layer so that they were all accessible from Caps mode.

## Part 6: Wrapping it Up!

Now we have built all of the layers and set each button target. The final “map” for my keyboard is as follows (the lines show layer changes, and the boxes show what happens after each key is pressed):

