

BT-CKP Firmware Guide

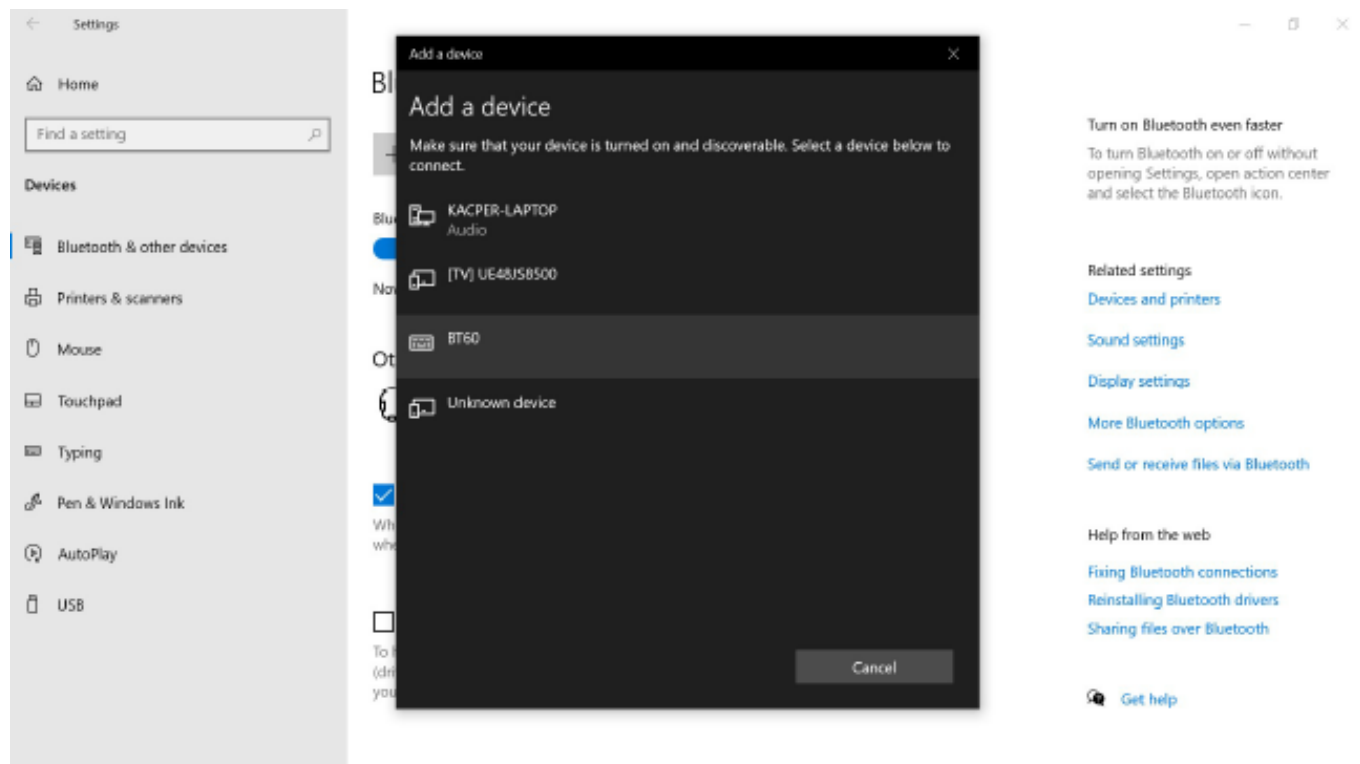
Welcome to your new keyboard on the Polarity Works Common Keyboard Platform. In this guide we will go through the process of pairing and customizing your firmware for your board.

Setting up your BT-CKP

Pairing

The BT60 is supplied powered on and ready to pair, if you bought more than one it's recommended to unplug the battery on all but one. Use a fingernail or small flathead screwdriver to gently pry the white battery connector out of the socket. DO NOT PULL THE PLUG BY THE WIRES as this can damage the wires/battery. We also recommend unplugging the battery prior to assembling the soldered unit. The connector is keyed and will only plug in one way, forcing it in the wrong way can damage both the connector and the circuitry. If soldering your own battery please ensure the polarity is correct, it is marked on the PCB with a plus sign.

The BT60 is compatible with BT4 or BT5 with BLE. Should your computer lack compatibility we recommend the TP Link UB4A. The drivers aren't great; however, each board was tested with it prior to shipping. To pair in Windows go to Settings - Devices - Add Bluetooth and other devices - Bluetooth. The keyboard advertises as "BT60". Click on it to pair, it should say "connecting" and a short while later it will say "Your device is ready to go". Hit some keys to make sure it's connected; it should appear with battery charge in the devices section. If you're having trouble check the troubleshooting section.



Charging

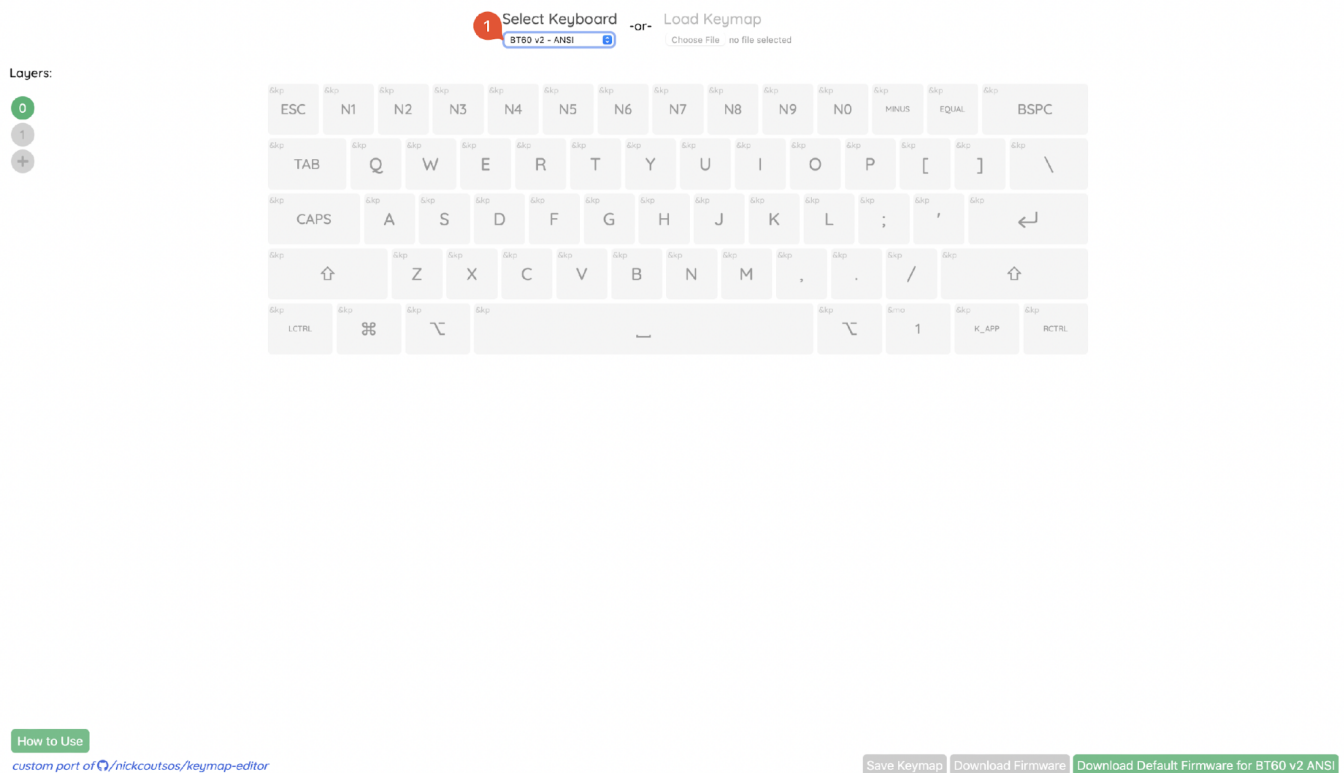
The BT60 charges whenever connected to USB so it's always ready for when you disconnect. There's a red LED on the underside of the board that illuminates when the battery is charging or when the battery is disconnected. It's dimly visible inside even in a metal case. A full charge should take about 3 hours using the included battery, and is expected to last over a month with the lights disabled. RGB and backlighting will shorten the battery life to days.

Customising your BT-CKP

Online Configurator

We have developed an online tool that allows you to edit the firmware without the need for any coding, you can find it here: <https://www.polarityworks.com/editor/index.html>

To get started, select your PCB from the dropdown menu. BT60 v1 firmware can be disregarded unless you have a black PCB.



1 Select Keyboard -or- Load Keymap
BT60 v2 - ANSI Choose File no file selected

Layers:

0
1
+

ESC N1 N2 N3 N4 N5 N6 N7 N8 N9 N0 MINUS EQUAL BSPC
TAB Q W E R T Y U I O P [] \
CAPS A S D F G H J K L ; ' ↵
↑ Z X C V B N M , . / ↵
LCTRL ⌘ ⌥ _ ⌥ 1 K_APP RCTRL

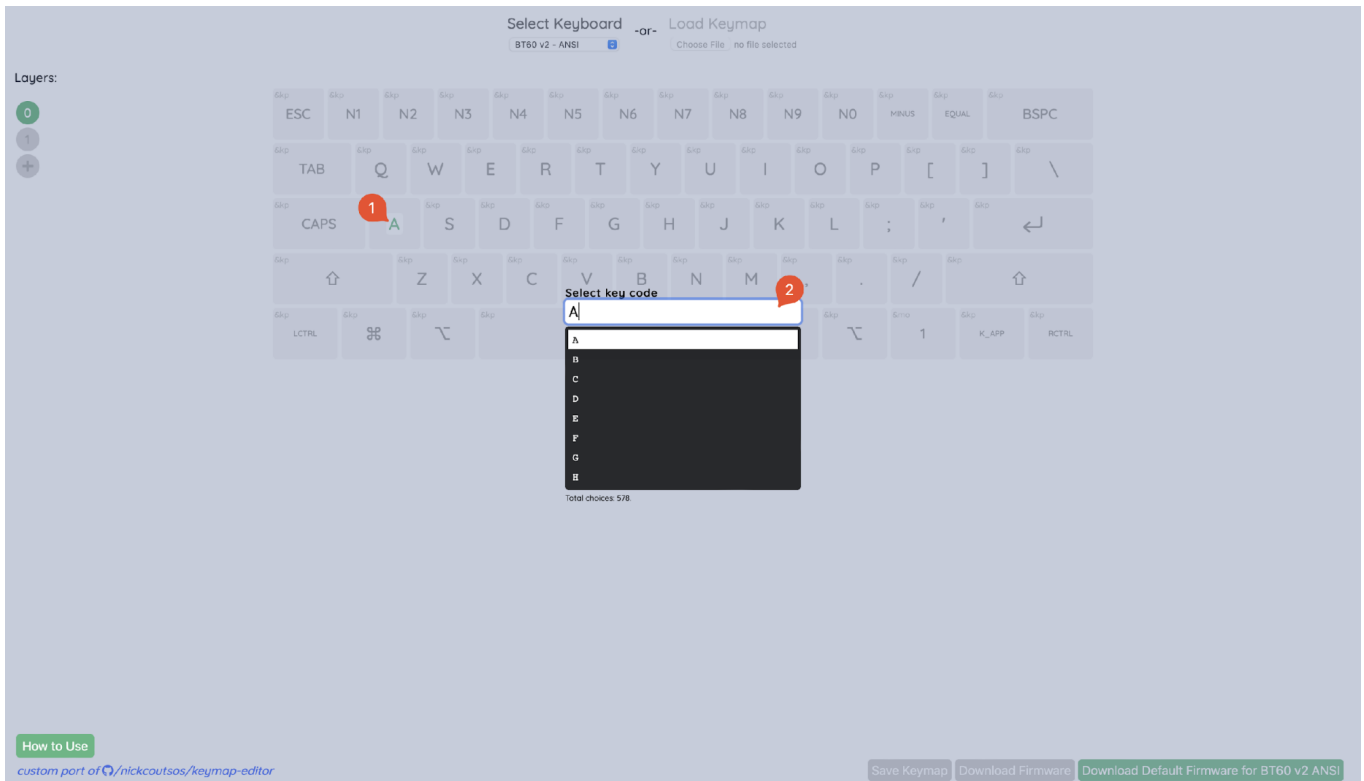
How to Use
custom port of [nickcoutsos/keymap-editor](#)

Save Keymap Download Firmware Download Default Firmware for BT60 v2 ANSI

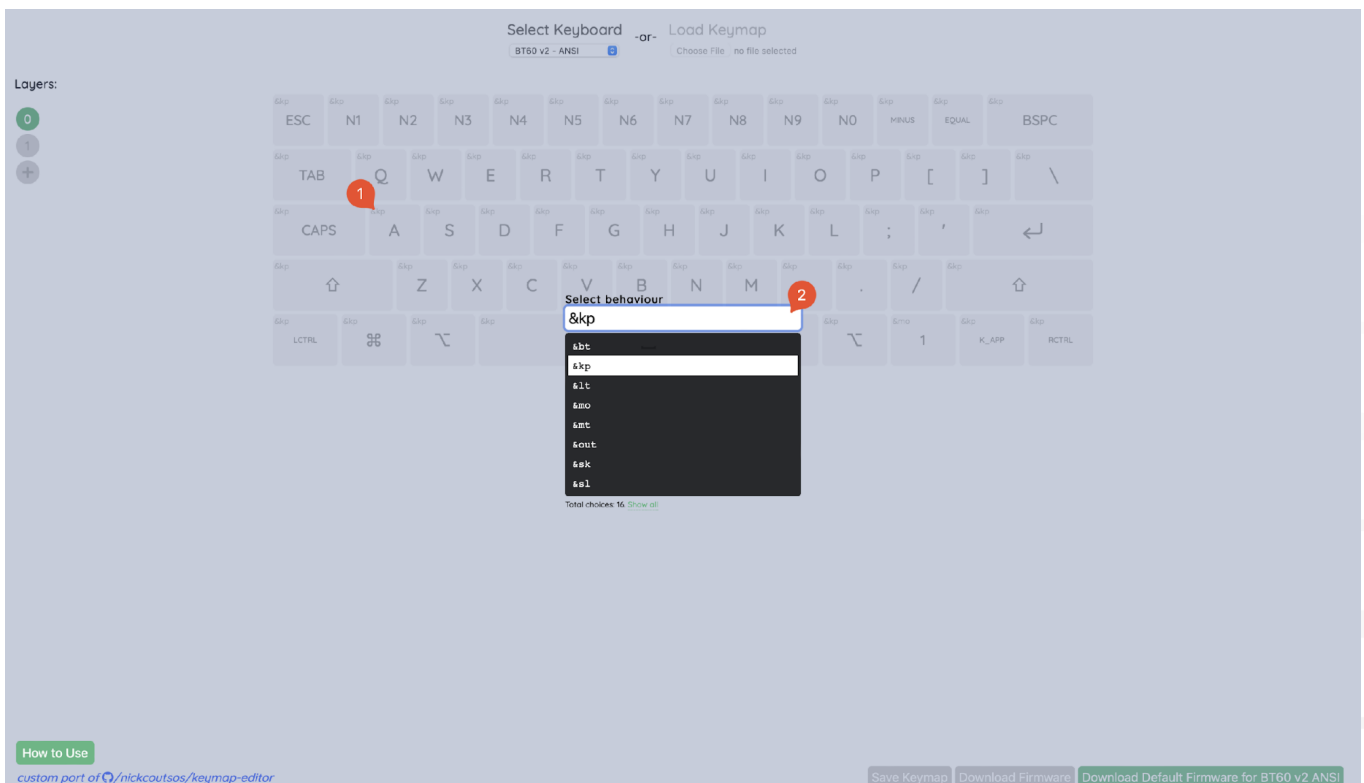
After selecting your PCB, the layout will change to match your PCB and you will have the option to download the default firmware or modify it. The “Download Firmware” button will be greyed out until you make a change.

POLARITY WORKS

To change the keycode for a given key, press the centre symbol on the key. This will bring up a search box, the full list of keycodes is available here: <https://zmk.dev/docs/codes>



You can change the key behaviours by clicking on the top left corner of the key, this will bring up a search bar you can use to find all the available behaviours. You can find out more about the different behaviours here: <https://zmk.dev/docs/behaviors/key-press>



POLARITY WORKS

On the left side of the screen, you will see the layers, with layer 0 being the base layer. Remember to bind a behaviour like `&mo` to switch to them from the base layer. To switch which layer is being displayed, press on its corresponding number. You can add up to 20 layers by pressing the plus button, to delete a layer press the cross that appears when you hover over it (be careful, this cannot be undone). You can rename each layer by double clicking on the layer name.

Select Keyboard

BT60 v2 - ANSI

-or-

Load Keymap

Choose File

no file selected

Layers:

0

1

2

+

| | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|
| ESC | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 | N0 | MENUS | EQUAL | BSPC |
| TAB | Q | W | E | R | T | Y | U | I | O | P | [|] | \ |
| CAPS | A | S | D | F | G | H | J | K | L | ; | ' | | ↵ |
| ⇧ | Z | X | C | V | B | N | M | , | . | / | | ⇧ | |
| LCTRL | ⌘ | ⌘ | | | | | | | | | 1 | ⌘ | RCTRL |

If the behaviour is changed the key will display a `0` symbol, this means the parameters of the behaviour (be it a key binding or other command) have not been set, if you try and build the firmware like this it will cause an error. The `&trans` behaviour can be used for keys you want to leave blank.

Setting Up

If you want to implement more advanced features than what's possible with the online GUI, you can edit the firmware code on GitHub. We're using the very versatile ZMK-config system, which allows us to rebuild custom firmwares without needing to setup any build environments, saving time, effort and disk space.

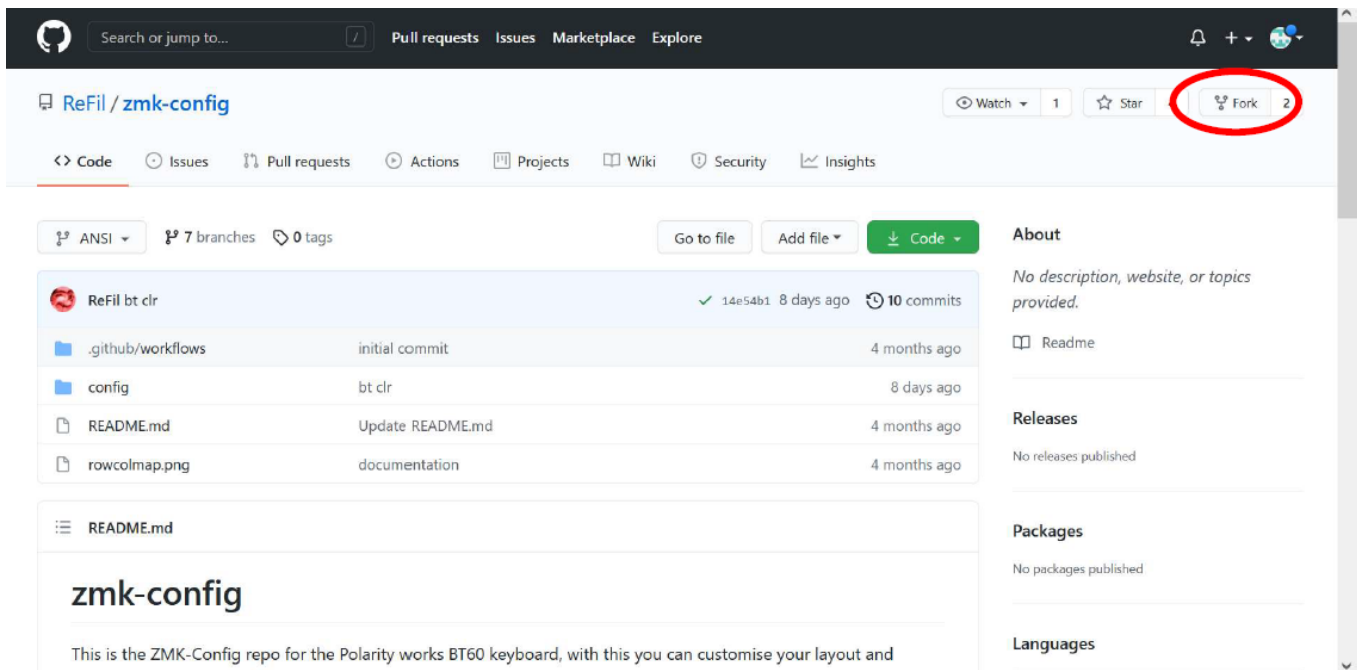
✦ Create a Github account

To setup a GitHub account you need only an email and to solve their CAPCHA puzzle. Visit this link to get to the signup screen: <https://github.com/signup>

✦ Fork the template repo we provide

We have example firmwares for several common other 60% layouts.

- link for V1 boards (Black PCB): <https://github.com/ReFil/zmk-config>
- link for V2, BT65 and BT75 boards (white PCB): <https://github.com/Polarity-Works/zmk-config-ckp>



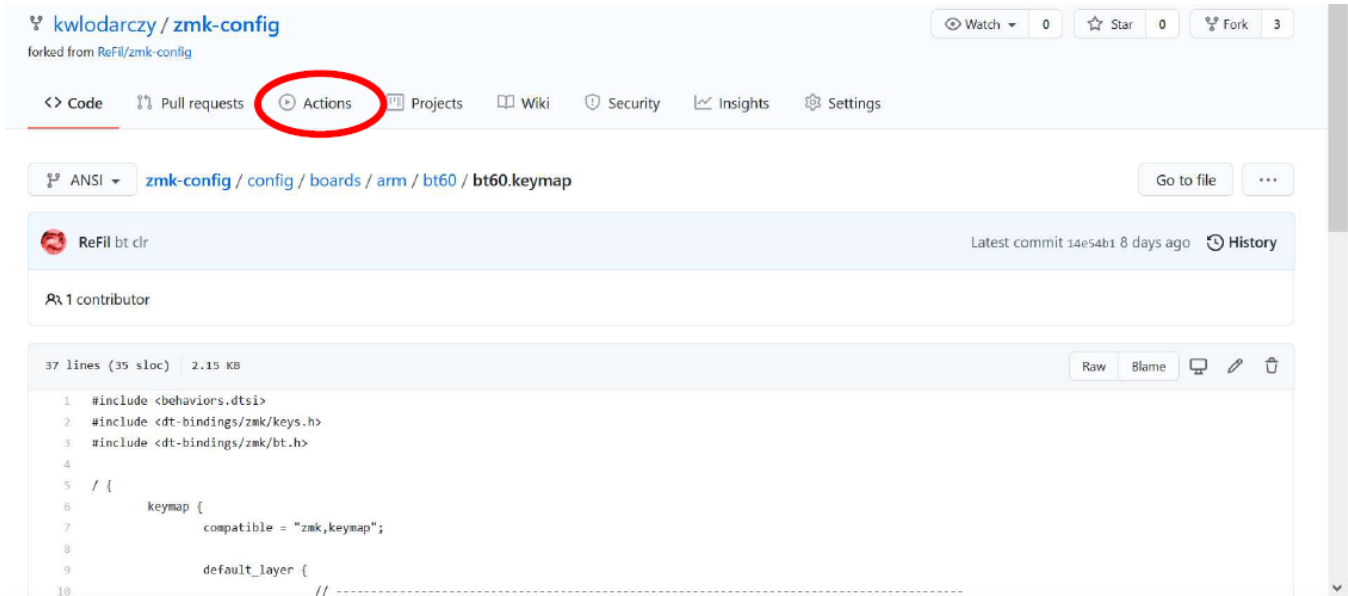
For the V1 boards: Proceed to click the drop-down menu that says "ANSI" and then select the layout of your choice or hotswap. Flashing the hotswap firmware to a soldered board or vice versa will lead to the keys being in the wrong place and you will need to reflash the keyboard.

For the V2 boards: Click the drop-down menu and select size and layout. Both hotswap and soldered V2 boards use the same firmware for the same layouts.

GitHub has a built-in text editor to handle online editing, so you don't need to download anything at all. Select the file you want to edit and press the button with the pencil icon.

Enable the build workflows

We have setup GitHub actions to automatically build the firmware in the cloud. Click on the "Actions" tab after saving changes and you should see a message saying, "Workflows aren't being run on this forked repository". Click the green "I understand" button to enable the automatic building actions.



Edit the Configuration

When altering the configuration of the BT60 there are two files that are important, the bt60.keymap contains the key bindings and the bt60.dts file which defines the physical layout of the keys. They can be found in config/boards/arm/bt60/bt60.keymap

When altering the configuration of the BT60 there are two files that are important, the bt60.keymap contains the key bindings and the bt60.dts file which defines the physical layout of the keys. They can be found in config/boards/arm/bt60

bt60.keymap

The bt60.keymap file contains all the layers with the keys they correspond to, if you have encoders set up you map them here too. ZMK uses a behaviours principle, whereby each physical keypress triggers a behaviour (denoted by &). Some of these behaviours trigger a keypress event (&kp), or an auxiliary control such as Bluetooth.

The full list of behaviours is documented here : <https://zmk.dev/docs>*! ZMK has all the key-codes documented here: <https://zmk.dev/docs/codes/>. Each layer needs to have the right number of keys to match the physical layout or else the build will fail.

— POLARITY WORKS +

bt60.dts

The bt60.dts file has the physical layout of the keyboard with other configuration options as well. The image in the repo dictates which columns and rows the keys connect to for the various layouts. The matrix transform documentation can be found here: https://zmk.dev/docs/development/new-shield/*optional-matrix-transform

To add new keys put the RC(X,Y) in the appropriate place in the matrix transform. Invalid keys might not fail to build but will cause unexpected behaviours.

When finished editing click the green “Save Changes” button. This will “commit” and “push” the changes you’ve made to your fork.

Building

Click on the “Actions” tab after saving changes and you should see a new action building. There will be an orange progress indicator. Click on it and you will see the progress of the building action. The build process typically takes 10-15 minutes. If the build succeeds you will see a green tick mark in the place of the orange dot, and if it fails there will be a red cross. Click on “Build” and you’ll see where it failed, typically in the “West Build (bt60)” section. In the first ~10 lines it’ll display the problem, including the line number in the file “bt60.dts.pre.tmp”. This file can be viewed in the drop down below west build (“bt60 DTS File”). Go to the relevant line number and you will usually see the problem, typically either an error in the behaviour definition (capitalisation is important!) or a keycode error. Edit the keymap or dts file to fix the problem and rebuild. Once the build succeeds, you’ll be able to download the firmware from the action summary page (it’s under the artifacts section).

✔ Update bt60.keymap Build #1

🔄 Re-run jobs ⋮

🏠 Summary

Jobs

✔ Build

Triggered via push 3 days ago

👤 kwłodarczy pushed → e153205 **AKST**

Status

Success

Total duration

3m 20s

Artifacts

1

build.yml

on: push

✔ Build

3m 3s

🔄 - +

Artifacts

Produced during runtime

Name

Size

📦 firmware

311 KB

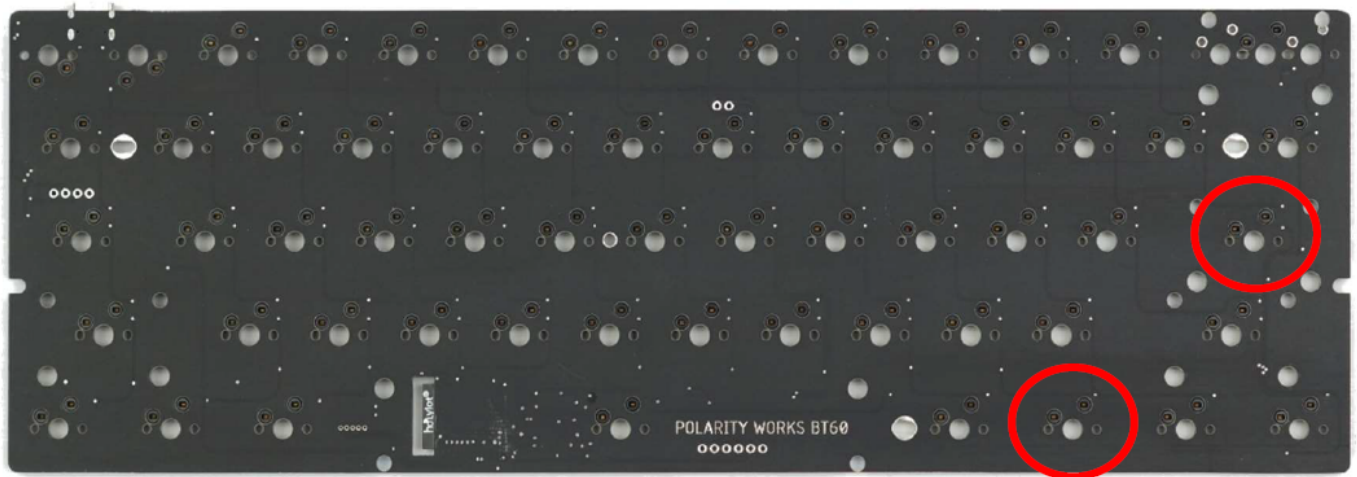
🗑



— POLARITY WORKS +

Flashing

Unzip the firmware.zip file and you will find a file called bt60.uf2. The bt60 uses a variant of the Adafruit nrf52 bootloader which allows software-less flashing. To get your bt60 into bootloader you can use a key combination or press the reset button on the underside of the board twice. On the default firmware on all boards you hold down the mod button (bottom row, third in from the right) and press the enter key.



At this point the green LED on the underside will start blinking, and the keyboard will show up in windows as a USB drive with the name "BT60" and three files on it. Drag and drop the bt60.uf2 file you downloaded onto the board and let it copy. Sometimes Windows will report an error message, but this doesn't necessarily mean the flashing has failed. The green led will extinguish once flashing has completed and the board will reappear as a keyboard. Paired devices are not forgotten between flashes, so don't unpair the keyboard on your computer after flashing it, or you'll need to clear Bluetooth devices.

Troubleshooting your BT-CKP

We tried to make the experience as painless as possible but inevitably some people may encounter issues. The best place to get in touch with us if this guide doesn't help is our discord. Follow this link: <https://discord.gg/6WNR94g69C>

Bluetooth

Bluetooth LE can sometimes be funky on Windows, occasionally you can get pairing and connectivity issues. If you get an issue pairing first step is to clear the Bluetooth profile on the keyboard, in ZMK this is known as BT_CLR. On the stock firmware for all variants it is bound to mod + RCTRL. We recommend shorting the pads with tweezers or a paperclip. After running BT_CLR you should ensure it is unpaired in Settings too. Try pairing again, sometimes you need to hit the clear button a few times or reboot the keyboard by either pressing the reset button (mod + \) or the physical button on the underside.

Flashing

We've yet to encounter issues, during the testing each board was flashed multiple times with different firmwares from several different computers. If you have an issue it is best to try and flash it again as its most likely a glitch. Other possible causes for flashing errors include a corrupted uf2 file (try flashing an unmodified firmware found on our website) or a dodgy USB cable (try a different one).

Customising

If you're having trouble with ZMK even when following the docs, you can reach out in our discord or join the official ZMK discord (<https://discord.com/invite/sycytVQ>) and ask there. The typical issues are slight typos in the keycodes or incorrect behaviours. Capitalisation is very important in the code.