adafruit learning system

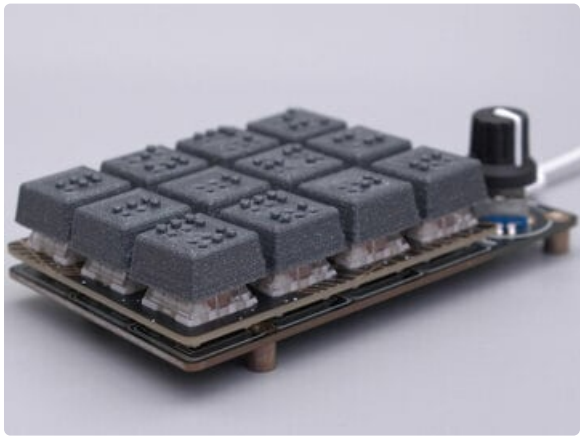
MacroPad Braille Keycaps

Created by Ruiz Brothers

Last updated on 2021-07-21 09:24:20 PM EDT

Guide Contents

Overview



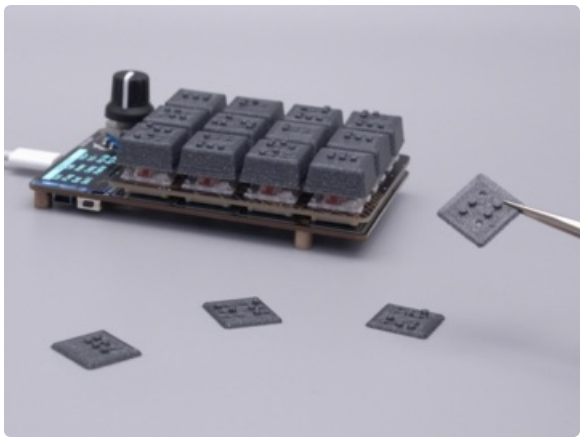
Accessibility Keycaps for the Mechanical Keyboards.

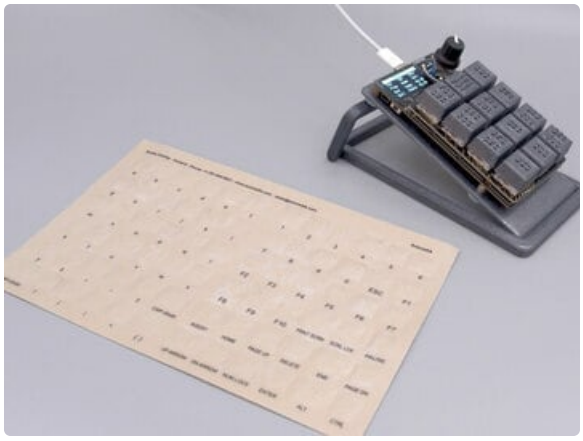
These keycaps are 3D printed with the braille alphabet so folks can touch and feel their keyboard shortcuts.

We thought it'd be nice to have audible feedback so a tone is played when a key is pressed.

The Adafruit MacroPad features the RP2040 chip and has a built-in speaker for generating tones and playing audio files.

We think the Adafruit MacroPad is great for making custom input devices for accessibility projects.





Braille is a writing system that features raised bumps called cells. To make these keycaps, we referenced a peel-and-stick overlay that features functions keys and special characters.

The keycaps are compatible with Cherry MX and Kailh Box switches.

You can print the keycaps in one piece or just the top cover for existing keycaps.

The keycaps are installed by press fitting them over the stem of the mechanical key switch.

The resolution and print quality is good enough for making this style of keycaps.

Parts

Adafruit MACROPAD RP2040 Bare Bones - 3x4 Keys + Encoder + OLED

Strap yourself in, we're launching in T-minus 10 seconds...Destination? A new Class M planet called MACROPAD! M here, stands for Microcontroller because this 3x4 keyboard...

Out of Stock

Out of Stock

Your browser does not support the video tag.

Adafruit MacroPad RP2040 Starter Kit - 3x4 Keys + Encoder + OLED

Strap yourself in, we're launching in T-minus 10 seconds...Destination? A new Class M planet called MACROPAD! M here stands for Microcontroller because this 3x4 keyboard...

Out of Stock

Out of Stock

USB Type C Cable with Data/Charge Switch

Perhaps your smart phone, tablet, or even your Raspberry Pi 4, charges off of USB, but can you really trust that charging station at the airport? Now you can control when to allow...

$4.95

In Stock

Add to Cart

Proto-Pasta - 1.75mm 500g Conductive PLA Filament

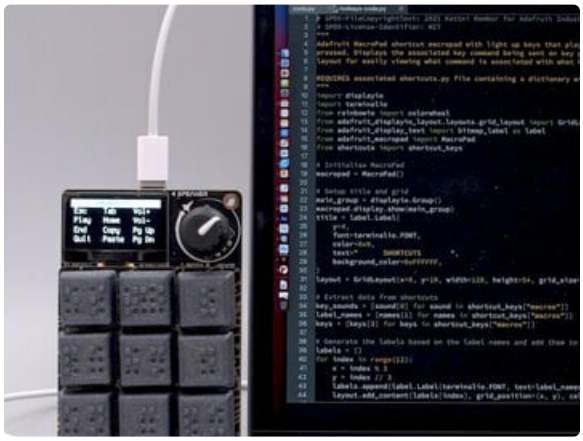
It'd be Fusilli not to buy this filament! Pici up this filament today! Did we...

$58.00

In Stock

Add to Cart

Code



This uses the CircuitPython MacroPad library to make customizable shortcuts with Lights and Sounds.

The main loop checks for key events and handles inputs if it's a keycode or consumer control such as volume and media playback.

The shortcuts are stored in a separate file that contains a dictionary of keys.

Here you can set the tone, label, keycode type and the key sequence.

With CircuitPython, you have access to the code with the USB drive, which is great when developing across different computers.

Code and Shortcuts

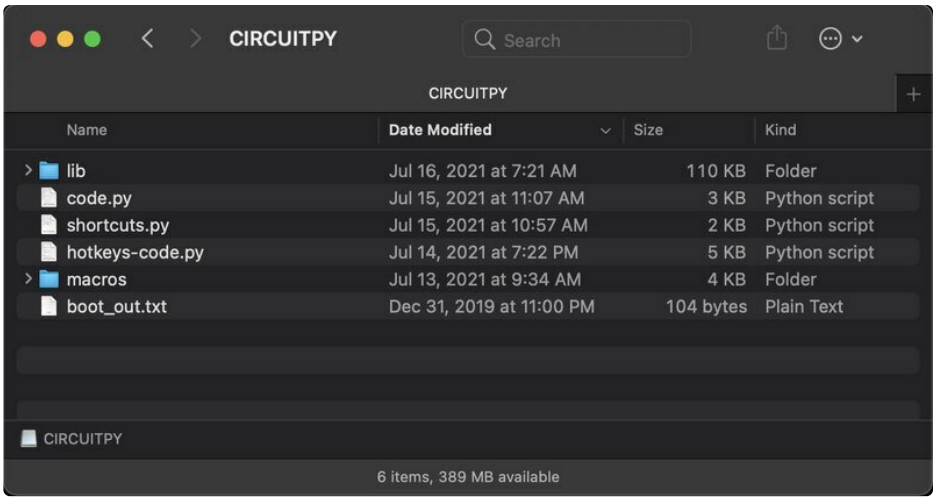
Follow the instructions to save the following **code.py** file, the **shortcuts.py** file, and the necessary libraries to your **CIRCUITPY** drive.

Click the **Download Project Bundle** button below to download the necessary libraries. the **code.py** file, and **shortcuts.py** file in a zip file. Extract the contents of the zip file and choose the folder that matches the version of CircuitPython you are using. Copy the **entire lib folder**, the **code.py** file, and the **shorcuts.py** file to your **CIRCUITPY** drive.

```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
# SPDX-License-Identifier: MIT
"""
Adafruit MacroPad shortcut macropad with light up keys that play a tone or wav file when a key
is
pressed. Displays the associated key command being sent on key press in a grid matching the key
layout for easily viewing what command is associated with what key.

REQUIRES associated shortcuts.py file containing a dictionary with all the key info.
"""
import displayio
```

```
import terminalio
from rainbowio import colorwheel
from adafruit_displayio_layout.layouts.grid_layout import GridLayout
from adafruit_display_text import bitmap_label as label
from adafruit_macropad import MacroPad
from shortcuts import shortcut_keys

# Initialise MacroPad
macropad = MacroPad()

# Setup title and grid
main_group = displayio.Group()
macropad.display.show(main_group)
title = label.Label(
    y=4,
    font=terminalio.FONT,
    color=0x0,
    text="      SHORTCUTS       ",
    background_color=0xFFFFFF,
)
layout = GridLayout(x=0, y=10, width=128, height=54, grid_size=(3, 4), cell_padding=5)

# Extract data from shortcuts
key_sounds = [sound[0] for sound in shortcut_keys["macros"]]
label_names = [names[1] for names in shortcut_keys["macros"]]
keys = [keys[3] for keys in shortcut_keys["macros"]]

# Generate the labels based on the label names and add them to the appropriate grid cell
labels = []
for index in range(12):
    x = index % 3
    y = index // 3
    labels.append(label.Label(terminalio.FONT, text=label_names[index], max_glyphs=10))
    layout.add_content(labels[index], grid_position=(x, y), cell_size=(1, 1))

# Display the text
main_group.append(title)
main_group.append(layout)

while True:
    key_event = macropad.keys.events.get()  # Begin checking for key events.

    if key_event:  # If there is a key event, e.g. a key has been pressed...
        if key_event.pressed:  # And a key is currently being pressed...

            # ... light up the pressed key with a color from the rainbow.
            macropad.pixels[key_event.key_number] = colorwheel(
                int(255 / 12) * key_event.key_number
            )

            # If it's a Keycode...
            if "KC" in shortcut_keys["macros"][key_event.key_number][2]:
                # ... send the associated key command or sequence of key commands.
                for key in keys[key_event.key_number]:
                    macropad.keyboard.press(key)
                macropad.keyboard.release_all()
```

```
            # If it's a ConsumerControlCode...
            if "CC" in shortcut_keys["macros"][key_event.key_number][2]:
                # ... send the associated consumer control code.
                for key in keys[key_event.key_number]:
                    macropad.consumer_control.send(key)

            sounds = key_sounds[key_event.key_number]  # Assign the tones/wavs to the keys.
            if isinstance(sounds, int):  # If the sound is a tone in Hz...
                macropad.start_tone(sounds)  # ... play the tone while the key is pressed.
            if isinstance(sounds, str):  # If the sound is a wav file name as a string...
                macropad.play_file(sounds)  # ... play the wav file.
        else:
            # Otherwise, turn off the NeoPixels and stop the tone.
            macropad.pixels.fill(0)
            macropad.stop_tone()
```



Configuration

The **shortcuts.py** file contains a Python dictionary that allows you to easily configure what each key will do in terms of the following:

- The tone in Hz to play.
- The label to show on the display (should be kept to 6 characters or less to fit).
- The key type being sent (e.g. KC for Keycode or CC for Consumer Control Code).
- The key sequence (the keycode or group of keycodes to send on key press).

Each of the multi-entry lines in the dictionary have all four pieces of information for each key. To configure a key differently, simply update the associated line to have the desired tone, display label, key type abbreviation, and the key sequence.

For example, to update the first key to play a different tone and send the letter "a", I would change line 29 to the following:

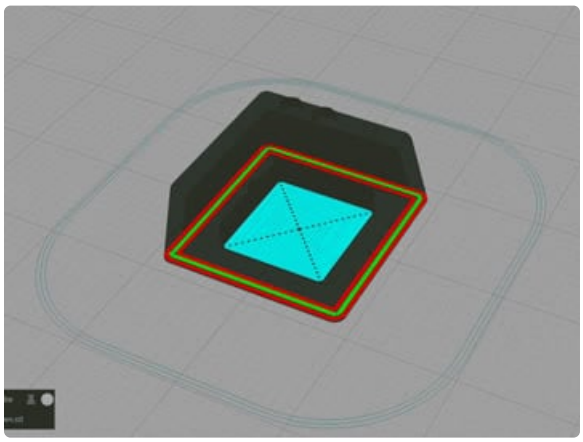
```
(185, 'a', 'KC', [macropad.Keycode.A]),
```

That's all there is to configuring your MacroPad shortcuts.

3D Printing

Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

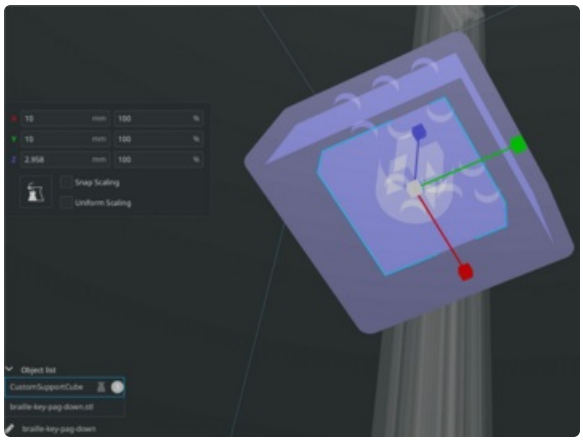


Slicing Parts

Supports are required for the full keycaps. Slice with setting for PLA material.

The parts were sliced using CURA using the slice settings below.

- PLA filament 220c extruder
- 0.2 layer height
- 10% gyroid infill
- 60mm/s print speed
- 60c heated bed



Supports

- Support Extrusion Width: .2
- Support Density: 4%
- Support Z Height: .21
- Interface: Off
- Support Roof: On
- Support Pattern: Zig Zag
- Support Roof Pattern: Zig Zag

https://adafru.it/TUF

https://adafru.it/TUF

https://adafru.it/TVF

https://adafru.it/TVF

Assembly

The keycaps are compatible with Cherry MX and Kailh Box switches.

You can print the keycaps in one piece or just the top cover for existing keycaps.

The keycaps are installed by press fitting them over the stem of the mechanical key switch.