# Table of Contents

## Overview
- Parts 3
- Keycaps 5
- M2.5 x 18mm Screws x3 6
- Tools and Materials 6

## Numpad Layout
- Getting Off the Grid 8
- Keyboard Layout Editor 8
- Plate & Case Builder 9
- Model 11

## Assemble the Numpad
- Snap 12
- Extra Column No More 13
- Snip Snip 14
- More Snip Snip 15
- Edge Cleanup 16
- 3D Case 17
- Keyswitches, Plate, PCBs 18
- One Offs 19
- Wiring 20
- Assembly 23
- Case Middle 25
- Case Top 27
- Feather Screws, Standoffs 28
- Case Bottom 29

## Install CircuitPython
- CircuitPython Quickstart 30
- Safe Mode 32
- Flash Resetting UF2 34

## Code and use the Numpad 4000
- Text Editor 35
- Download the Project Bundle 35
- Use the Numpad 4000 37
Overview

Compact keyboards are super cool -- I use a [TKL](https://adafruit.it/UgE) -- but sometimes you just miss having a big, old number pad for data entry and calculations. The Numpad 4000 is just that! Plus, you can move it off to either side of your main keyboard for ideal ergonomics.

You can build the custom mechanical Number Pad of your Dreams with the Ortho NeoKey Snap-Apart PCB, a Feather RP2040, and CircuitPython. Customize the physical layout on this diode-matrixed, NeoPixel-lit wonder board!

Parts

NeoKey 5x6 Ortho Snap-Apart Mechanical Key Switches w/ NeoPixel
For folks who want ready-to-go keeb action, we’ve got the lovely Adafruit Macropad with a 3x4 grid of MX+NeoPixel key switches -...
[https://www.adafruit.com/product/5157](https://www.adafruit.com/product/5157)
This project should work well on nearly any CircuitPython-capable Feather board. I chose to use the very lovely Feather RP2040.

**Black Adafruit Feather RP2040**
A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought “this chip is going to be awesome when we give it the Feather...”
https://www.adafruit.com/product/4884

**Key Switches and Keycaps**

The NeoKey Ortho 5x6 PCB uses up to 30 keyswitches and key caps, although for the Numpad 4000 you’ll only need 22.

**Kailh Mechanical Key Switches - Thick Click Jade Box - 10 pack**
For crafting your very own custom keyboard, these Kailh Clicky Jade mechanical key switches are deeeee-luxe! With smooth actuation and Cherry MX...
https://www.adafruit.com/product/5149

**Kailh Mechanical Key Switches - 10 packs - Cherry MX Compatible**
For crafting your very own custom keyboard, these Kailh mechanical key switches are deeeee-luxe! Come in a pack of 10 switches, plenty to make a...
https://www.adafruit.com/product/4996
Keycaps

To build the Numpad 4000 as designed here you'll need a set of number pad-specific keys. I used SA profile, double-shot Maxkey keycaps (https://adafru.it/UgA) -- these numberpad keys happened to be left over from a TKL (ten key-less) build I did. Talk to a friend who has ordered some fancy keycaps before -- in many cases they'll have left over numpad keys as people tend to make smaller boards!

Or, go for a fully orthographic build using 1u keycaps such as these DSA profile ones:

Black DSA Keycaps for MX Compatible Switches - 10 pack
Dress up your mechanical keys in your favorite colors, with a wide selection of stylish DSA key caps. Here is a 10 pack of Adafruit black DSA keycaps for your next mechanical keyboard...
https://www.adafruit.com/product/4997

Woven USB Cable with USB Type A to Right Angle USB Type C
Some days we're feeling extra fancy here at the 'fruit warehouse, and we have a big soft spot for woven fabric cables. Like, peep
https://www.adafruit.com/product/5031

Black Nylon Machine Screw and Stand-off Set – M2.5 Thread
Totaling 380 pieces, this M2.5 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel your maker...
https://www.adafruit.com/product/3299
Little Rubber Bumper Feet - Pack of 4
Keep your electronics from going barefoot, give them little rubber feet!
These small sticky bumpers are our favorite accessory for any electronic kit or device. They are sticky, but...
https://www.adafruit.com/product/550

M2.5 x 18mm Screws x3
These ones (https://adafruit.it/UgB) from McMaster-Carr are nice.

Tools and Materials

You'll need a 3D printer and filament to build the switch plate and case. Alternately, you can make a variant stacked design using acrylic or wood on a laser cutter or mill, or even with chipboard and a CNC cutter such as the Cricut.

You'll also need a soldering iron, solder, and thin wire such as this:

Silicone Cover Stranded-Core Wire - 30AWG in Various Colors
Silicone-sheathing wire is super-flexible and soft, and its also strong! Able to handle up to 200°C and up to 600V, it will do when PVC covered wire wimps out. We like this wire...
https://www.adafruit.com/product/2051
Numpad Layout

Not all keycaps are square. Keycap ratios are expressed as proportional widths relative to a "one unit", or, "1u" standard, which you find on most of the "normal" alphanumeric keys on a keyboard.

Other key sizes exist: modifiers are usually 125% width, or 1.25u. Here are some common sizes (these may vary with different designs):

- "normal" keys = 1u
- alt, ctrl = 1.25u
- tab = 1.5u
- caps lock = 1.75u
- numpad 0, +, enter = 2u
- spacebar = 6.25u
Here's a nice resource for learning more about keyboard anatomy, [Keyboard University](https://adafru.it/Uga).

Getting Off the Grid

Since this design uses a few 2u keycaps, we'll need to move some of the NeoKey snap-apart PCBs off of the default 1u grid.

The NeoKey Ortho 6x5 Snap-Apart PCB makes it easy to rearrange the layout.

To keep things neat and stable, you can design a key plate for 3D printing, laser cutting, or CNC milling.

An excellent tool for designing your layout in the browser is the [Keyboard Layout Editor](https://adafru.it/Ugb).

Keyboard Layout Editor

An excellent tool for designing your layout in the browser is the [Keyboard Layout Editor](https://adafru.it/Ugb).

You can start a new, blank layout, then add keys and rearrange their sizes and positions using the per-key controls.

Using the Keyboard Layout Editor I created the layout I want.
Once the layout is set, I headed to the Raw Data tab to copy the markup text shown here:

```javascript
[{c: "#606cc4", t: "ffffff", p: "DSA", a: 5, f: 4}, "NUM


UP", {c: "#408dff", a: 7, f: 9}, "7", "+", "9", {c: "#606cc4", h: 2, s: 0}, "+"], [{a: 5, f: 3}, "PAGE


```

Highlight and copy the markup text so you can use it in the next step.

Note, I added the _s:0 code to remove stabilizer cutouts from the 2u switches.

**Plate & Case Builder**

Now that you have the keyboard laid out, you can use the [swillkb Plate & Case Builder](https://adafru.it/Ugc) to generate the CAD drawings of your plate layout.

Paste the markup text from the Raw Data you copied in the Keyboard Layout Editor in to the Plate Layout field. Pick and options you want, and then click the Draw My CAD!!! button.
Pick the CAD Output tab and you'll see your drawing.

Click on the file type button to download an SVG, DXF, or EPS file of the drawing. You'll use this in your 3D modeling package of choice, or to generate a laser cutter or CNC toolpath.

numpad4k.svg

https://adafruit.it/Ugd
Model

Most 3D modeling and CAD programs will allow you to import the drawing file of your choice. I used the venerable .dxf file and imported it into Rhino/Grasshopper.

I then created a Grasshopper workflow to effectively extrude the collection of curves 1.6mm in height to generate the switch plate model. (I also had it offset the outer boundary curve by 2mm and fillet the corners.)

The 1.6mm plate height is a good starting point, allowing the switches to click into place nicely, but you can try making it thicker if you like.

Export the model as an .stl file and 3D print it! (Note, the print shown here was made using the notched keyswitch profile, but the squares work just as well or better due to increased connection surface.)
Assemble the Numpad

Snap
The first step is to snap off the top and bottom of the PCB.

Use some pliers to gently bend at the perforation until the extra material snaps off.

Please use eye protection when breaking apart printed circuit boards.
Extra Column No More
Pry off the extra column at the far right of the board, as seen from the top side of the PCB.

You can save these key PCBs for use in another project later, such as a five-key macro strip!
Snip Snip
Use diagonal cutters to cut off the four key PCBs of the fifth column as shown, taking care to leave the first PCB in place.

Then, snap off two of the four key PCBs you just removed and reserve them for use with the + and ENTER keys later in the build.

You will eventually wire and solder these loose PCBs back into the numpad matrix.
More Snip Snip
Repeat the previous process to turn the bottom row of four into a spaced out row of three that will accommodate the 2u spacing of the 0 key.
This is the fundamental numpad layout you'll be working with. Next we'll clean up the board edges.

PCBs contain fiberglass which is hazardous to your health. Wear a facemask when filing or sanding the edges.

**Edge Cleanup**

Use your diagonal cutters to remove some of the excess material where the boards were separated, then file them down with a small metal file or sandpaper.

Be sure not to breathe in the dust!
3D Case

Use the files linked below and print the plate, top, base, and bottom parts.

Numpad 4000 Case Files

https://adafruit.it/Ugf
Keyswitches, Plate, PCBs
Arrange the PCBs as shown, then snap a few keyswitches into place.

Be careful to align the two metal legs of the keyswitch with the sockets of the PCBs.
One Offs
For the 2u spaced keys, snap the keyswitches through the plate, then press the PCBs on from the back.
Wiring

It's time to wire it all up! There are two sets of wiring tasks here -- one is to connect the key matrix column and row pins to the Feather RP2040 as well as the power, ground and NeoPixel pin. The other wiring task is to re-connect the snapped-off key PCBs to the others.

Note how the NeoPixel data line runs in a snake-like pattern through the grid.

Follow this wiring diagram to make the connections.
Here the plate and keyswitches have been removed to expose the wiring more clearly.

Note: Sometimes the PCB traces that run between the key PCBs can become damaged when neighboring PCBs have been cut, just due to strain on the thin connection. You can repair these with a short jumper wire as shown on the third PCB on the bottom row in the following photograph.
Assembly
Now that you’ve wired everything, you can insert all of the keyswitches, and then add the keycaps.
Case Middle
Feed the Feather RP2040 through the case middle section as shown.
Case Top
Place the case top on top of the boards so you can sandwich everything and screw the parts together.

You can start with the three long screws.
Feather Screws, Standoffs
Add the four shorter screws to the Feather mounting holes.

Then, thread the seven hex standoffs onto the screws.
Case Bottom
Set the case bottom in place and then use the smallest screws to attach it to the hex standoffs from below.

You can also add rubber bumper feet as shown for a non-skid experience.
The Numpad 4000 is assembled and ready for coding in CircuitPython.

**Install CircuitPython**

[CircuitPython](https://adafru.it/tB7) is a derivative of [MicroPython](https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

**CircuitPython Quickstart**

Follow this step-by-step to quickly get CircuitPython running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org

[https://adafru.it/R1D](https://adafru.it/R1D)
Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.

To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
You will see a new disk drive appear called RPI-RP2.

Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.

The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.
Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

Entering Safe Mode in CircuitPython 6.x

To enter safe mode when using CircuitPython 6.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 700ms. On some boards, the onboard status LED (highlighted in green above) will turn solid yellow during this time. If you press reset during that 700ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

Entering Safe Mode in CircuitPython 7.x

To enter safe mode when using CircuitPython 7.x, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will...
blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

Once you've entered safe mode successfully in CircuitPython 6.x, the LED will pulse yellow.

If you successfully enter safe mode on CircuitPython 7.x, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

Auto-reload is off.
Running in safe mode! Not running saved code.

CircuitPython is in safe mode because you pressed the reset button during boot.
Press again to exit safe mode.

Press any key to enter the REPL. Use CTRL-D to reload.

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After loading this UF2, follow the steps above to re-install CircuitPython.

Download flash erasing "nuke" UF2

https://adafruit.it/RLE
Code and use the Numpad 4000

Text Editor

Adafruit recommends using the Mu editor for editing your CircuitPython code. You can get more info in this guide (https://adafruit.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the code.py file, along with a folder full of key configuration files. To get everything you need, click on the Download Project Bundle link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your Feather board's CIRCUITPY drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

```python
import board
import keypad
import neopixel
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode

COLUMNS = 5
ROWS = 5

BLUE = 0x000510
WHITE = 0x303030
RED = 0xFF0000

board_pix = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.1)
board_pix[0] = BLUE

key_pixels = neopixel.NeoPixel(board.D5, 30, brightness=0.1)
key_pixels.fill(WHITE)

keys = keypad.KeyMatrix(
    row_pins=(board.D4, board.A3, board.A2, board.A1, board.A0),
    column_pins=(board.D13, board.D12, board.D11, board.D10, board.D9),
    columns_to_anodes=False,
)

kbd = Keyboard(usb_hid.devices)

keycode_LUT = [
```
```python
pixel_LUT = [
  0, 1, 2, 3, 4,
  5, 6, 7, 8,
  9, 10, 11, 12, 13, 14,
  15, 16, 17, 18,
  19, 20, 21, 22, 23, 24
]

pixel_LUT = [
  0, 1, 2, 3, 4,
  5, 6, 7, 8,
  9, 10, 11, 12, 13, 14,
  15, 16, 17, 18,
  19, 20, 21, 22, 23, 24
]

# create a keycode dictionary including modifier state and keycodes
keymap = {
    0: (0, Keycode.KEYPAD_NUMLOCK),
    1: (0, Keycode.BACKSPACE),
    2: (0, Keycode.FORWARD_SLASH),
    3: (0, Keycode.KEYPAD_ASTERISK),
    4: (0, Keycode.KEYPAD_MINUS),
    5: (0, Keycode.PAGE_UP),
    6: (0, Keycode.KEYPAD_SEVEN),
    7: (0, Keycode.KEYPAD_EIGHT),
    8: (0, Keycode.KEYPAD_NINE),
    9: (0, Keycode.PAGE_DOWN),
   10: (0, Keycode.KEYPAD_FOUR),
   11: (0, Keycode.KEYPAD_FIVE),
   12: (0, Keycode.KEYPAD_SIX),
   13: (0, Keycode.KEYPAD_PLUS),
   14: (1, Keycode.SHIFT),
   15: (0, Keycode.KEYPAD_ONE),
   16: (0, Keycode.KEYPAD_TWO),
   17: (0, Keycode.KEYPAD_THREE),
   18: (2, Keycode.CONTROL),
   19: (0, Keycode.KEYPAD_ZERO),
   20: (0, Keycode.KEYPAD_PERIOD),
   21: (0, Keycode.KEYPAD_EQUALS)  # KEYPAD_ENTER on non-mac
}

keymap = {
    0: (0, Keycode.KEYPAD_NUMLOCK),
    1: (0, Keycode.BACKSPACE),
    2: (0, Keycode.FORWARD_SLASH),
    3: (0, Keycode.KEYPAD_ASTERISK),
    4: (0, Keycode.KEYPAD_MINUS),
    5: (0, Keycode.PAGE_UP),
    6: (0, Keycode.KEYPAD_SEVEN),
    7: (0, Keycode.KEYPAD_EIGHT),
    8: (0, Keycode.KEYPAD_NINE),
    9: (0, Keycode.PAGE_DOWN),
   10: (0, Keycode.KEYPAD_FOUR),
   11: (0, Keycode.KEYPAD_FIVE),
   12: (0, Keycode.KEYPAD_SIX),
   13: (0, Keycode.KEYPAD_PLUS),
   14: (1, Keycode.SHIFT),
   15: (0, Keycode.KEYPAD_ONE),
   16: (0, Keycode.KEYPAD_TWO),
   17: (0, Keycode.KEYPAD_THREE),
   18: (2, Keycode.CONTROL),
   19: (0, Keycode.KEYPAD_ZERO),
   20: (0, Keycode.KEYPAD_PERIOD),
   21: (0, Keycode.KEYPAD_EQUALS)  # KEYPAD_ENTER on non-mac
}

shift_mod = False
ctrl_mod = False

while True:
    key_event = keys.events.get()
    if key_event:
        if key_event.pressed:
            if keymap[keycode_LUT.index(key_event.key_number)][0] == 1:
                shift_mod = True
            elif keymap[keycode_LUT.index(key_event.key_number)][0] == 2:
                ctrl_mod = True
            if shift_mod is False and ctrl_mod is False:
                kbd.press(keymap[keycode_LUT.index(key_event.key_number)][1])
                print(keymap[keycode_LUT.index(key_event.key_number)][1])
                key_pixels[pixel_LUT.index(key_event.key_number)] = RED
            elif shift_mod is True and ctrl_mod is False:
                kbd.press(Keycode.SHIFT,
                          keymap[keycode_LUT.index(key_event.key_number)][1])
                print(keymap[keycode_LUT.index(key_event.key_number)][1])
                key_pixels[pixel_LUT.index(key_event.key_number)] = RED
            elif shift_mod is False and ctrl_mod is True:
                kbd.press(Keycode.CONTROL,
                          keymap[keycode_LUT.index(key_event.key_number)][1])
                print(keymap[keycode_LUT.index(key_event.key_number)][1])
                key_pixels[pixel_LUT.index(key_event.key_number)] = RED
            else:
                kbd.press(Keycode.SHIFT,
                          keymap[keycode_LUT.index(key_event.key_number)][1])
                print(keymap[keycode_LUT.index(key_event.key_number)][1])
                key_pixels[pixel_LUT.index(key_event.key_number)] = RED
```
Use the Numpad 4000

Once the libraries and code are installed, the Numpad 4000 will work as a USB HID keyboard device. You can try it out right away by plugging the keyboard into your computer via a known good USB cable and then typing in some numbers and symbols.

This guide page (https://adafruit.it/DaD) has a great intro to CircuitPython HID Keyboard.

For even more details, check out the documentation at https://circuitpython.readthedocs.io/projects/hid/en/latest/ (https://adafruit.it/B-7) which includes all of the keycodes and media codes you can use.

If you want to customize the keys, to send different keycodes, this dictionary contains all of the mappings:

```python
keymap = {
    (0): (0, Keycode.KEYPAD_NUMLOCK),
    (1): (0, Keycode.BACKSPACE),
    (2): (0, Keycode.FORWARD_SLASH),
    (3): (0, Keycode.KEYPAD_ASTERISK),
    (4): (0, Keycode.KEYPAD_MINUS),
    (5): (0, Keycode.PAGE_UP),
    (6): (0, Keycode.KEYPAD_SEVEN),
    (7): (0, Keycode.KEYPAD_EIGHT),
    (8): (0, Keycode.KEYPAD_NINE),
    (9): (0, Keycode.PAGE_DOWN),
    (10): (0, Keycode.KEYPAD_FOUR),
    (11): (0, Keycode.KEYPAD_FIVE),
    (12): (0, Keycode.KEYPAD_SIX),
```

(13): (0, Keycode.KEYPAD_PLUS),
(14): (1, Keycode.SHIFT),
(15): (0, Keycode.KEYPAD_ONE),
(16): (0, Keycode.KEYPAD_TWO),
(17): (0, Keycode.KEYPAD_THREE),
(18): (2, Keycode.CONTROL),
(19): (0, Keycode.KEYPAD_ZERO),
(20): (0, Keycode.KEYPAD_PERIOD),
(21): (0, Keycode.KEYPAD_EQUALS)  # KEYPAD_ENTER on non-mac