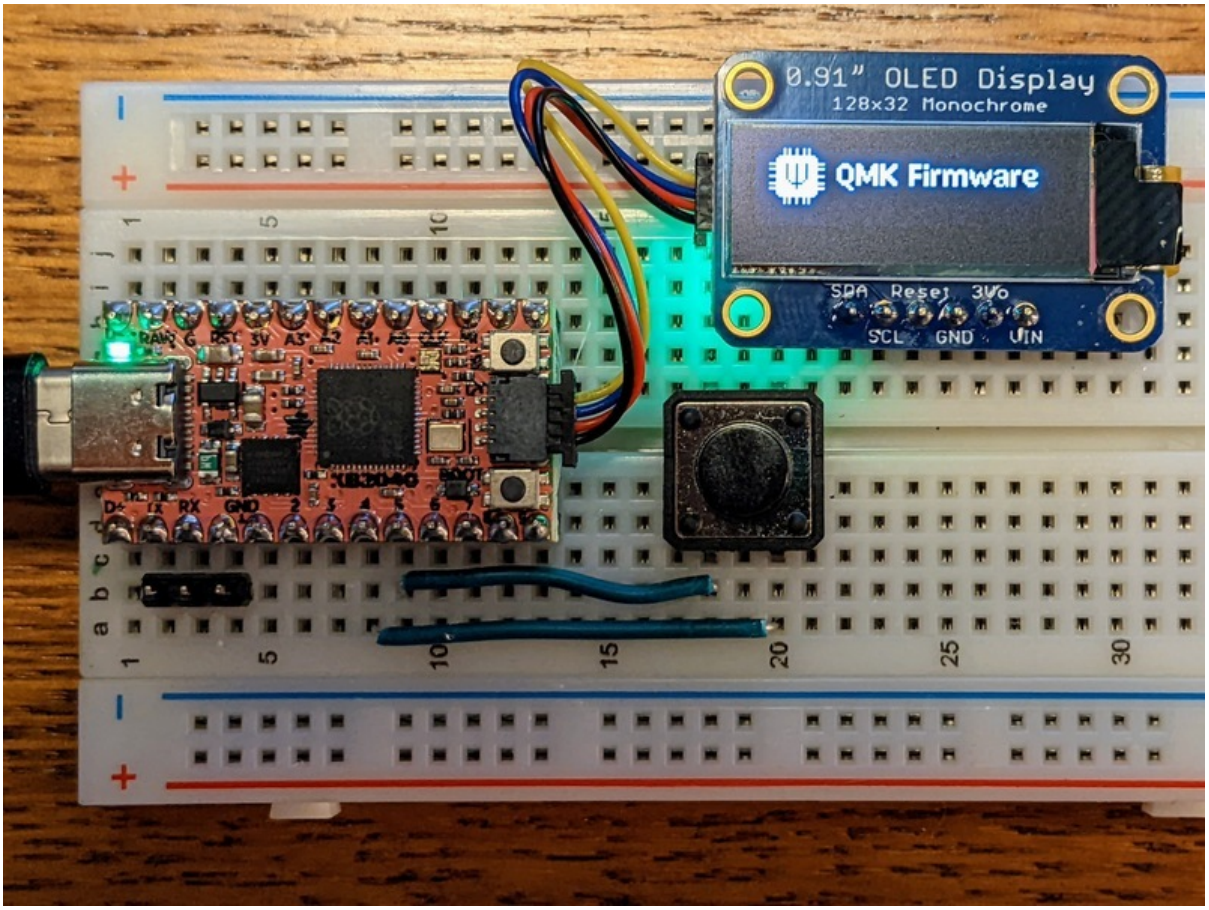
Using QMK on RP2040 Microcontrollers

Created by Jeff Epler

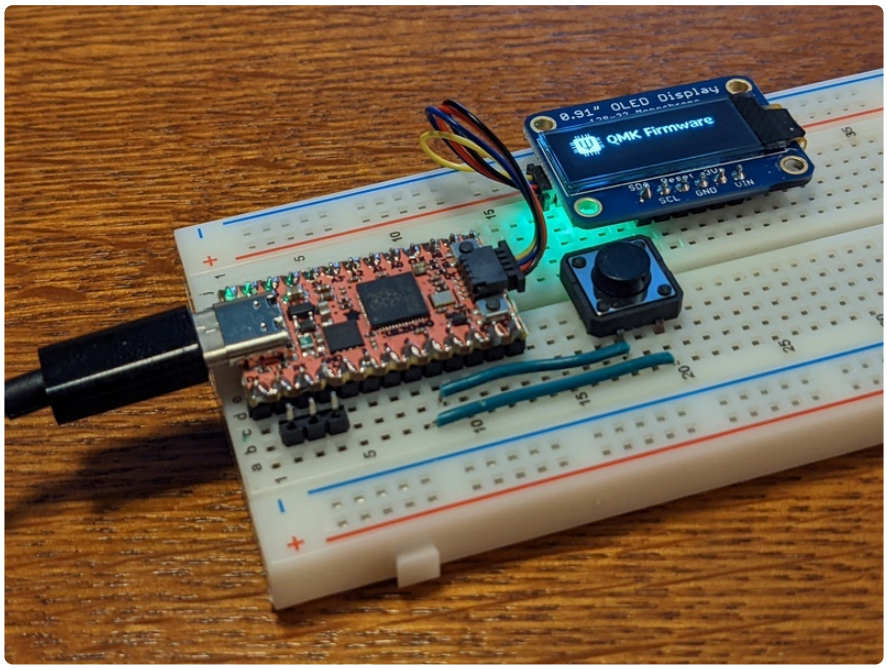


https://learn.adafruit.com/using-qmk-on-rp2040-microcontrollers

Last updated on 2024-06-03 03:39:21 PM EDT

Table of Contents

Overview



Love mechanical keyboards? You're among friends. Friends who want to show you how to use QMK with the great RP2040 microcontroller from Raspberry Pi. Support for this was added in July 2022 to the **develop** branch by the QMK community.

If you don't care to set up your computer to compile QMK, each example also includes a **.uf2** file that you can download and immediately start using on a compatible RP2040 board, no software installation required.

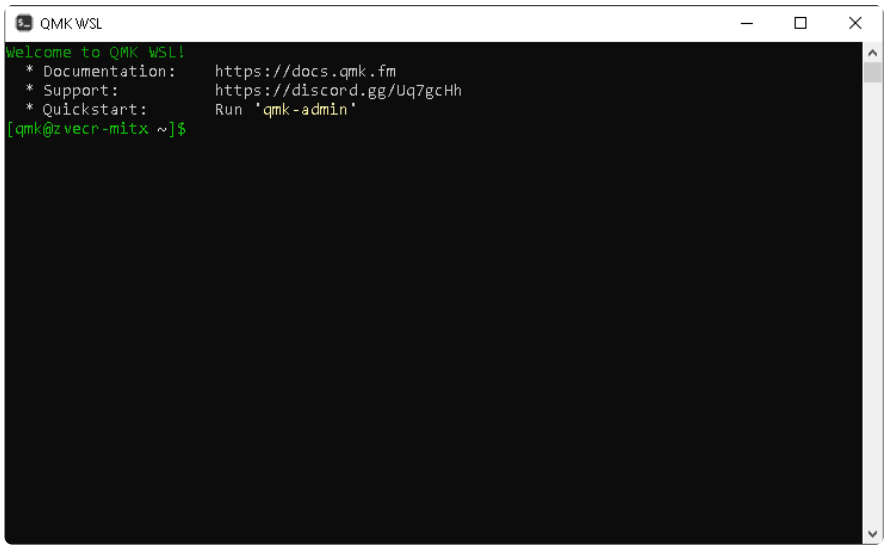
Install QMK on your computer

Each kind of operating system has different steps for this, and if you run into trouble check the official instructions (https://adafru.it/10dw) from qmk for more information and troubleshooting info!

Windows-based systems

For Windows, the best practice is to install "QMK WSL (https://adafru.it/11dA)" following the official instructions. If you prefer, you can install the older "QMK MSYS (https://adafru.it/10dx)" instead, but in general QMK WSL is faster and more reliable for compiling QMK code.

Heading 3 Delete Cancel Save



Debian-based systems including Ubuntu

For Debian-based Linux distributions (including Ubuntu and Raspberry Pi OS), install git and pip using the package manager and then install the qmk command:

```
sudo apt install -y git python3-pip
python3 -mpip install --user qmk
```

For macOS systems

For macOS, you will need to install Homebrew. Follow the instructions on https://brew.sh (https://adafru.it/OLE). Then, install the `qmk` command using brew:

```
brew install qmk/qmk/qmk
```

Finish setting up the qmk environment

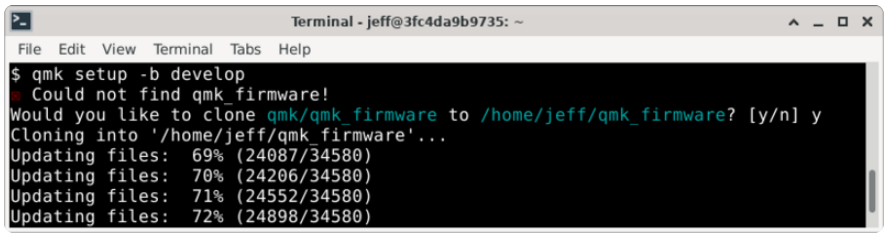
Check that the `qmk` command is working for you by asking it to report its version:

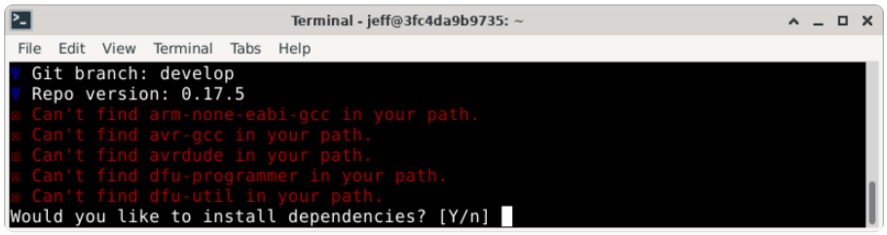
```
$ qmk --version
1.1.0
```

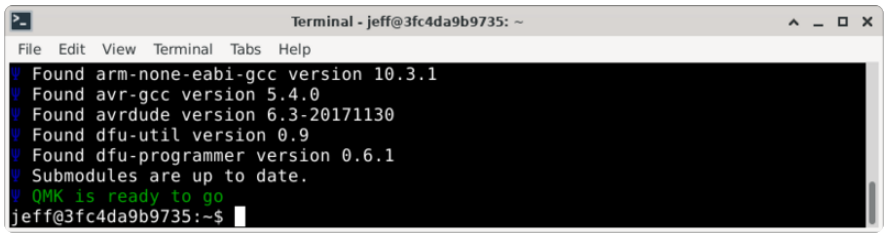
Next, for any kind of system, run

```
qmk setup
```

Answer "y" when asked to clone the firmware and install dependencies. Depending on your system, this may also prompt you for your password to install additional operating system packages.







Once this is done, you will have the `qmk` command available, and an up to date copy of the source code in **qmk_firmware** folder within your home folder. The following pages of this guide show how to build and install a firmware.

Switching back to CircuitPython or Arduino

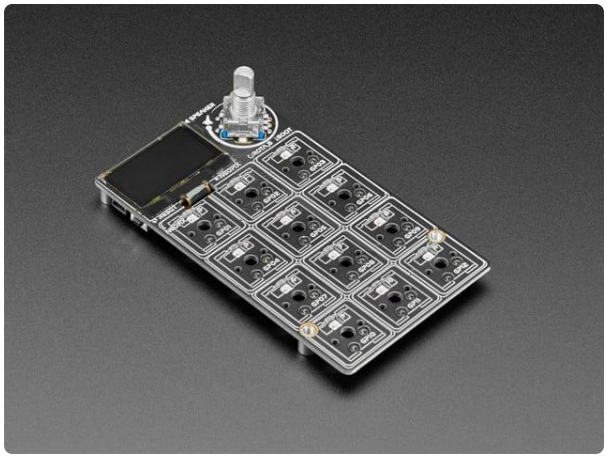
Your RP2040 is fully reprogrammable through the RPI-RP2 bootloader, but automatic reset to upload from the Arduino environment does not work when the board has a QMK firmware installed. Instead, you need to manually access the bootloader. For more details on how to access the bootloader, check the guide for your particular board. Here are some general instructions:

- If a RESET button is available, double click it. This works only when the firmware is correctly functioning. If it doesn't work, use one of the below methods.
- If a RESET button is available, hold down the BOOT button and click RESET once.
- Hold down the BOOT button while plugging in the device

Once your computer has recognized the RPI-RP2 bootloader, you can either drag a new UF2 to the drive (CircuitPython, QMK, or Arduino) or initiate the build and upload of a firmware (Arduino or `qmk flash`)

Parts

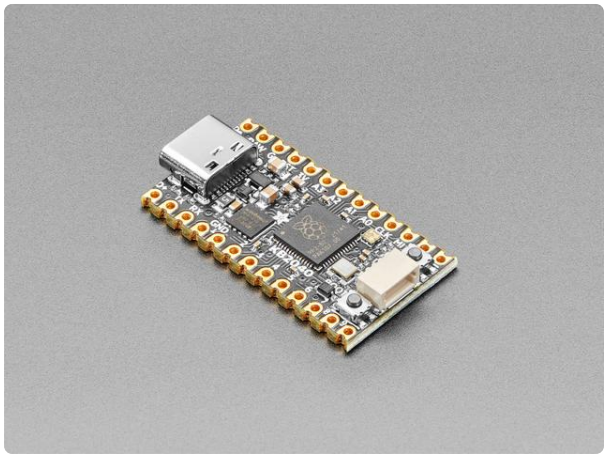
Ready to get down to business? Pick up some compatible hardware, then continue to one of the other pages in this guide to flash the QMK firmware to your RP2040 board, and learn more about how to modify and customize QMK along the way. If you don't feel like compiling software right now, we've also provided a **.uf2** file for each example so you can load it on your board right away.



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...

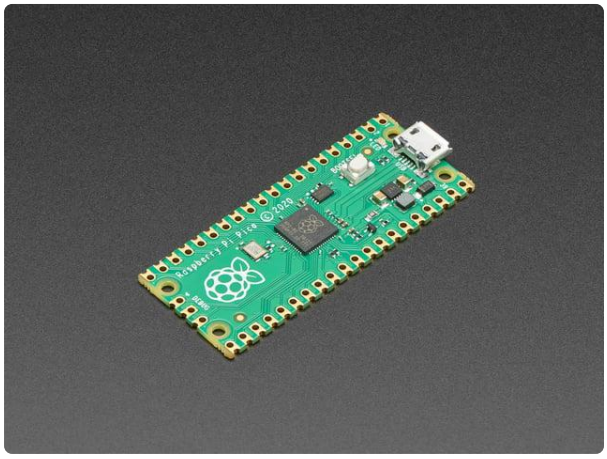
https://www.adafruit.com/product/5100



Adafruit KB2040 - RP2040 Kee Boar Driver

A wild Kee Boar appears! It's a shiny KB2040! An Arduino Pro Micro-shaped board for Keebs with RP2040. (#keeblife 4 evah) A lot of folks like using Adafruit...

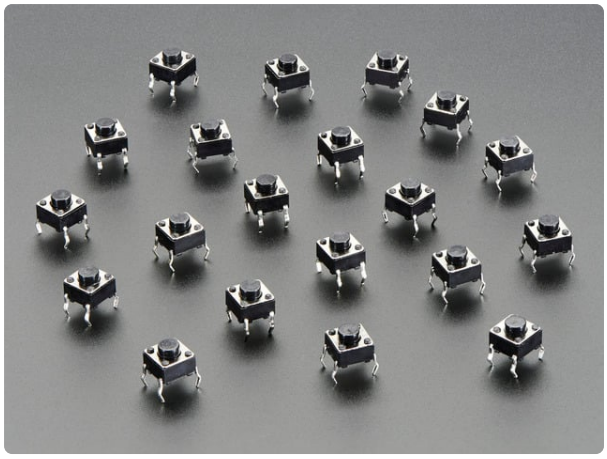
https://www.adafruit.com/product/5302



Raspberry Pi Pico RP2040

The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...

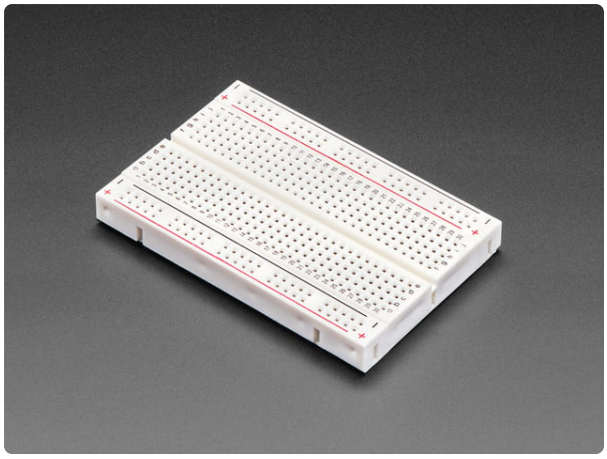
https://www.adafruit.com/product/4864



Tactile Button switch (6mm) x 20 pack

Little clicky switches are standard input "buttons" on electronic projects. These work best in a PCB but

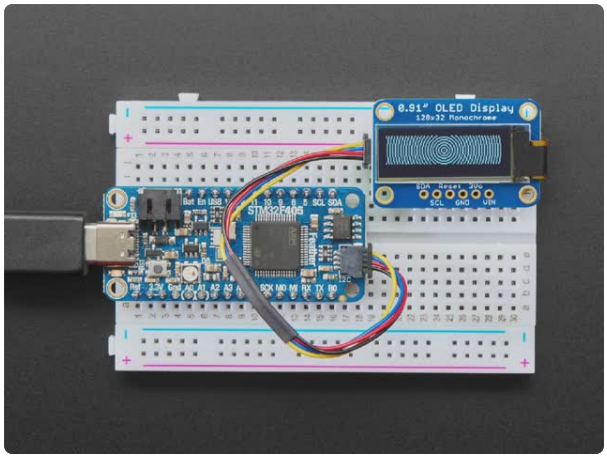
https://www.adafruit.com/product/367



Half Sized Premium Breadboard - 400 Tie Points

This is a cute, half-size breadboard with 400 tie points, good for small projects. It's 3.25" x 2.2" / 8.3cm x 5.5cm with a standard double-strip in the...

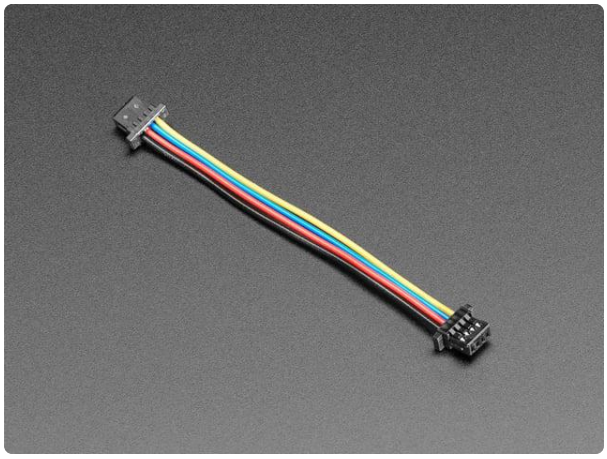
https://www.adafruit.com/product/64



Monochrome 0.91" 128x32 I2C OLED Display - STEMMA QT / Qwiic

These displays are small, only about 1" diagonal, but very readable due to the high contrast of an OLED display. This display is made of 128x32 individual white OLED pixels, each...

https://www.adafruit.com/product/4440

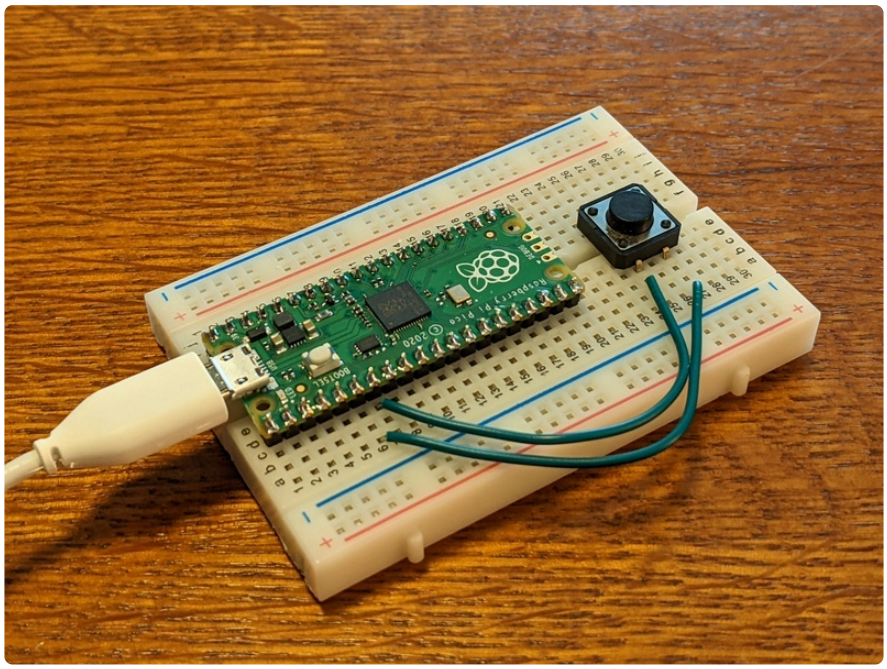


STEMMA QT / Qwiic JST SH 4-Pin Cable - 50mm Long

This 4-wire cable is 50mm / 1.9" long and fitted with JST SH female 4-pin connectors on both ends. Compared with the chunkier JST PH these are 1mm pitch instead of 2mm, but...

https://www.adafruit.com/product/4399

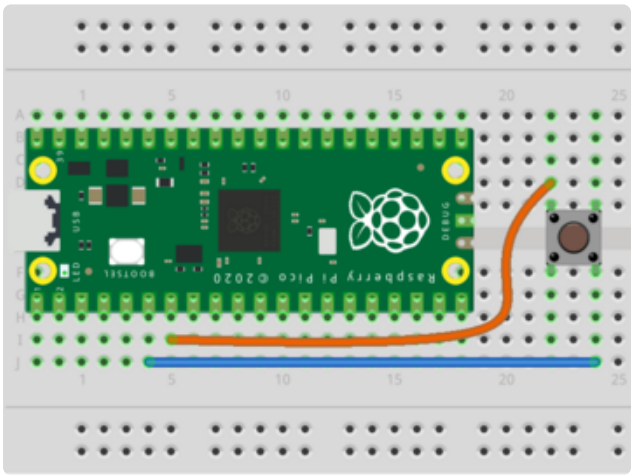
RP2040 One-Key Keyboard



Some keyboard definitions are designed for the RP2040. A simple example is the "onekey" keyboard. You can easily test this on a breadboard with a Raspberry Pi Pico and a simple button, so it's a great way to get started. This guide shows how to build the firmware file yourself, but if you want to take a short cut you can use the uf2 file below:

One-Key UF2 for RP2040

https://adafru.it/10dy



Connect **GP4** to one terminal of the button
Connect **GP5** to the diagonally opposite terminal of the button

Now, open up the terminal/command window, and change to the **qmk_firmware** directory.

Next, plug in you Pico while holding down the BOOT button so that the RPI-RP2 drive appears.

To build and install the firmware, issue the following command:

```
cd ~/qmk_firmware
qmk flash -kb handwired/onekey/rp2040 -km default
```

This will produce a lot of output. The start and end should look something like this:

```
~/qmk_firmware$ qmk flash -kb handwired/onekey/rp2040 -km default
Ψ Compiling keymap with gmake --jobs=1 handwired/onekey/rp2040:default:flash


QMK Firmware 0.17.5
Making handwired/onekey/rp2040 with keymap default and target flash

arm-none-eabi-gcc (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.2.1 20201103
(release)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Generating: .build/obj_handwired_onekey_rp2040/src/
info_config.h                                      [OK]
…[many lines skipped]
Copying handwired_onekey_rp2040_default.uf2 to qmk_firmware
folder                                  [OK]

Size after:
   text    data     bss     dec     hex filename
      0   30824       0   30824    7868 handwired_onekey_rp2040_default.uf2

Flashing /media/jepler/RPI-RP2 (RPI-RP2)
Wrote 61952 bytes to /media/jepler/RPI-RP2/NEW.UF2
```

The Raspberry Pi Pico will automatically restart with the new firmware. Your computer may notify you that a new keyboard has been connected. When you press the button, it will type the letter "a".

To change the key, you can open the file **qmk_firmware/keyboards/handwired/onekey/keymaps/default/keymap.c**. Just change `KC_A` to any valid key. For instance, to make it a mute button, use `KC_KB_MUTE` — generally, this special key will mute or unmute the computer's main sound output. QMK has a Frequently Asked Questions page about keycodes and keymaps (https://adafru.it/10dw) as well as a full list of all keycodes (https://adafru.it/10dw), so head over there to learn more about the possibilities.

```
#include QMK_KEYBOARD_H

const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
    LAYOUT_ortho_1x1(KC_A) // Try changing KC_A to KC_KB_MUTE
};
```

Adafruit MacroPad with QMK

Adafruit's MacroPad sports an RP2040 microcontroller, and a community member contributed support for it to QMK! This makes it easy to compile and install a QMK firmware on your MacroPad. If you just want to try out QMK without compiling anything, take a short-cut and use this pre-compiled uf2 file:

QMK for Adafruit MacroPad

https://adafru.it/10dz

Plug in your MacroPad while pressing down on the encoder knob, so that it enters boot mode and your computer recognizes the RPI-RP2 boot drive.

Open up the terminal/command window, and change to the **qmk_firmware** directory.

In this case the keyboard is **adafruit/macropad** and the keymap is **default**. Knowing this, you can issue the command to build and flash the firmware:

```
cd ~/qmk_firmware
qmk flash -kb adafruit/macropad -km default
```

This will produce a lot of output, similar to the following:

```
~/qmk_firmware$ qmk flash -kb adafruit/macropad -km default -c
QMK Firmware 0.17.5
Making adafruit/macropad with keymap default and target clean

Ψ Compiling keymap with gmake --jobs=1 adafruit/macropad:default:flash


QMK Firmware 0.17.5
Making adafruit/macropad with keymap default and target flash

arm-none-eabi-gcc (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.2.1 20201103
(release)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Generating: .build/obj_adafruit_macropad/src/
info_config.h                                          [OK]
…[Lots of lines omitted]
Copying adafruit_macropad_default.uf2 to qmk_firmware
folder                                       [OK]

Size after:
   text    data     bss     dec     hex filename
      0   47664       0   47664    ba30 adafruit_macropad_default.uf2

Flashing /media/jepler/RPI-RP2 (RPI-RP2)
Wrote 95744 bytes to /media/jepler/RPI-RP2/NEW.UF2
```

Your MacroPad will automatically restart with the QMK firmware. Simply press the keys on the MacroPad to type the digits 0-9, enter, and backspace. Rotate the encoder knob to increase/decrease system volume, and click the encoder knob in to mute/unmute.

To customize the layout, you can edit the file **adafruit/macropad/keymaps/default/keymap.c**. Look for this part of the file:

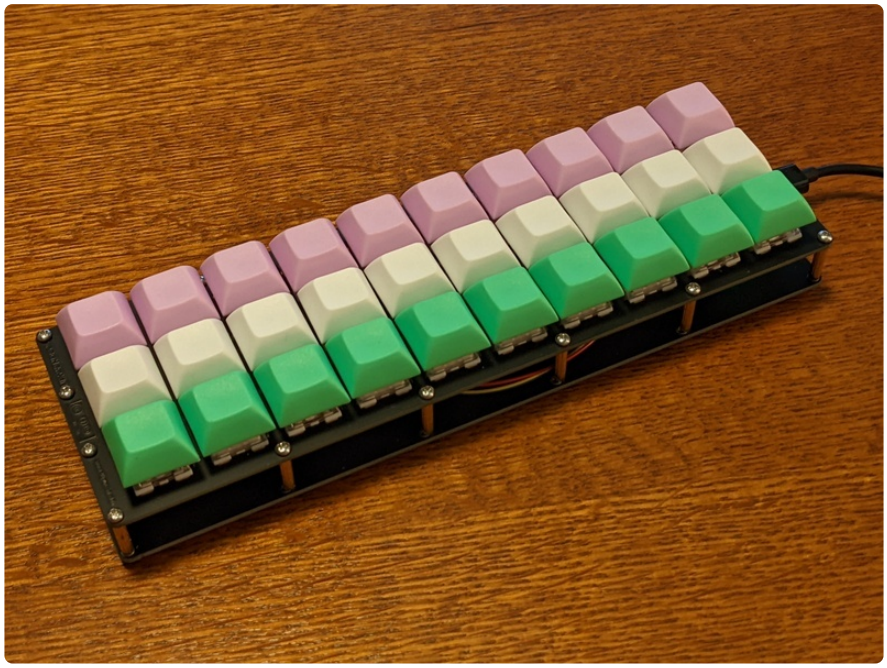
```
const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
  [0] = LAYOUT(
                    KC_MUTE,
      KC_ENT, KC_0, KC_BSPC,
      KC_7,   KC_8, KC_9,
      KC_4,   KC_5, KC_6,
      KC_1,   KC_2, KC_3
  )
};

#ifdef ENCODER_MAP_ENABLE
const uint16_t PROGMEM encoder_map[][NUM_ENCODERS][2] = {
  [0] = { ENCODER_CCW_CW(KC_VOLU, KC_VOLD) },
};
#endif
```

For instance, to change the layout to be more like a telephone dial pad than a computer numeric keypad, just re-arrange the rows of the keymap like so:

```
const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
  [0] = LAYOUT(
                    KC_MUTE,
      KC_1,   KC_2, KC_3
      KC_4,   KC_5, KC_6,
      KC_7,   KC_8, KC_9,
      KC_ENT, KC_0, KC_BSPC,
  )
};
```

There's a lot more to know about QMK keycodes, so start with with their Keymap FAQ (https://adafru.it/10dw). You can go really deep here—for instance, layers (https://adafru.it/10dw) are a powerful way to use the same physical key to perform different functions depending on the currently selected layer.

Adafruit KB2040 on the PB Gherkin 30% Keyboard



The PB Gherkin is a "30% Keyboard", indicating that it has 30 keys (about 30% of the number of keys on a classic PC keyboard). This can be a challenging keyboard to use, since the keyboard layout is organized in "layers". However, it's a comparatively inexpensive way to enter the world of hand-soldered custom keyboards!

This keyboard is designed for a "pro micro" microcontroller board. The Adafruit KB2040 is in the same form factor, and QMK can automatically convert a "pro micro" keyboard firmware to work on the KB2040.

Assembly of this keyboard requires soldering skills and the instructions must be followed closely. You can find our assembly instructions in this dedicated guide (https://adafru.it/10dA). It's highly recommended to get the keyboard working with CircuitPython first before switching to QMK.

Don't feel like building the firmware yourself? You can give QMK a quick try by dragging the UF2 file below to the RPI-RP2 bootloader drive:

QMK for Gherkin with KB2040

https://adafru.it/10dB

The build steps similar to for the other firmware we've followed in this guide. The keyboard is **40percentclub/gherkin** and the keymap is **default**. This time, since the

firmware was originally designed for the pro micro, we have to specify that QMK should "convert to" the KB2040 using the `CONVERT_TO` setting. Just add that to the command line when making the firmware:

```
cd ~/qmk_firmware
qmk flash -kb 40percentclub/gherkin -km default -e CONVERT_TO=kb2040
```

In either case, you should get a build log displayed in the terminal, and at the end a .uf2 file is created:

```
Ψ Compiling keymap with gmake --jobs=1 40percentclub/gherkin:default:flash


QMK Firmware 0.17.5
Making 40percentclub/gherkin with keymap default and target flash

arm-none-eabi-gcc (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.2.1 20201103
(release)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Size before:
   text    data     bss     dec     hex filename
      0   32256       0   32256    7e00 40percentclub_gherkin_default_kb2040.uf2

Compiling: keyboards/40percentclub/gherkin/
gherkin.c                                                [OK]
Compiling: .build/obj_40percentclub_gherkin/src/
default_keyboard.c                                  [OK]
[many lines deleted]
Linking: .build/
40percentclub_gherkin_default_kb2040.elf
[OK]
Creating binary load file for flashing: .build/
40percentclub_gherkin_default_kb2040.bin             [OK]
Creating load file for flashing: .build/
40percentclub_gherkin_default_kb2040.hex                    [OK]
Creating UF2 file for deployment: .build/
40percentclub_gherkin_default_kb2040.uf2                   [OK]
Copying 40percentclub_gherkin_default_kb2040.uf2 to qmk_firmware
folder                              [OK]

Size after:
   text    data     bss     dec     hex filename
      0   32240       0   32240    7df0 40percentclub_gherkin_default_kb2040.uf2

Flashing /media/jepler/RPI-RP2 (RPI-RP2)
Wrote 64512 bytes to /media/jepler/RPI-RP2/NEW.UF2
```

The Default Keymap

The first layer contains the standard alphabet, escape, backspace, space, and enter. To access further features, hold one or more keys on the bottom row.

For instance, to type an uppercase "Z", hold the "Enter" key (which functions as shift when held) and press the "Z" key.

The second image is "layer 1", so to type a "0" hold down "Space" and press "P". The symbols that go with shifted numbers also appear on "layer 2" without shift, so to type a ")", hold down "Space", "Enter", and press "P" (Layer1+Shift) OR hold down "Backspace" and press "P" (Layer2).

Use further layers to access punctuation, cursor movement keys, and operating system special keys.

The symbol ▽ indicates that this layer doesn't define the key, so the base layer's definition is used. So for example, holding C and pressing L types ";" but holding C and pressing S types "s".

For more on layers, see the dedicated page in QMK's documentation (https://adafru.it/10dw).

Editing a keymap with QMK Configurator

You can edit a QMK keymap online with the QMK Configurator, then build and flash locally to customize the keymap according to your preference.

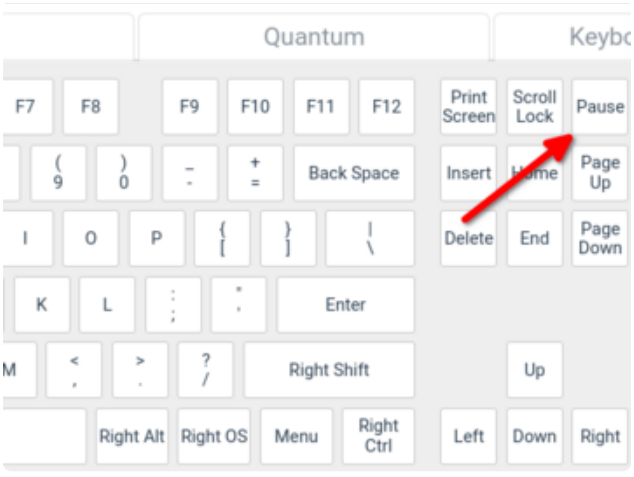
First, open this link to the configurator: https://config.qmk.fm/#/40percentclub/gherkin/LAYOUT_ortho_3x10 (https://adafru.it/10dC) and make any changes you like. For instance, if you use the "Pause" key to active your computer's lock screen or screen saver, you will find that it is missing on the default layout. Let's add it to Layer

5 in the position of the Enter key, so that holding B (layer 5) and pressing Enter will send the Pause key to the computer.

Select "Layer 5", then click the key in the lower right corner.

On the image of a full-size keyboard, click the Pause key.

Now the function of this key within this layer has been changed to "Pause".



Near the top of the page, click the button to download the keymap as a json file.

Import the keymap file into **qmk_firmware**:

```
cd ~/qmk_firmware/keyboards/40percentclub/gherkin
qmk import-keymap ~/Downloads/40percentclub_gherkin_layout_ortho_3x10_mine.json
```

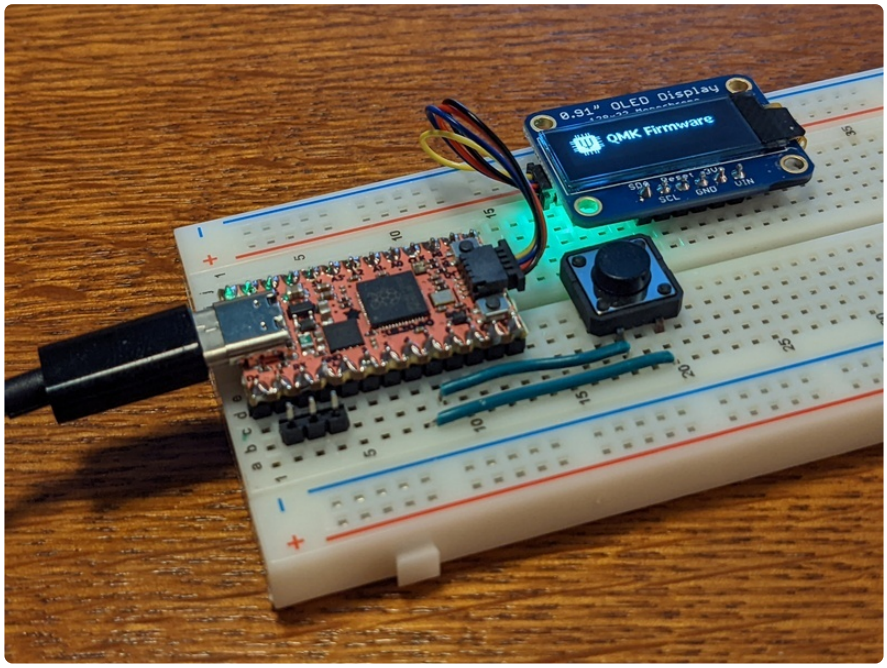
This should produce output similar to:

```
Ψ Importing 40percentclub_gherkin_layout_ortho_3x10_mine.json.

Ψ Imported a new keymap named 40percentclub_gherkin_layout_ortho_3x10_mine.
Ψ To start working on things, `cd` into keyboards/40percentclub/gherkin/keymaps/
40percentclub_gherkin_layout_ortho_3x10_mine,
Ψ or open the directory in your preferred text editor.
Ψ And build with qmk compile -kb 40percentclub/gherkin -km
40percentclub_gherkin_layout_ortho_3x10_mine.
```

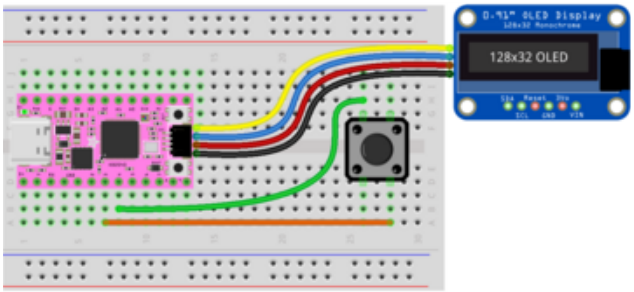
Now let's build and flash it:

```
cd ~/qmk_firmware
qmk flash -e CONVERT_TO=kb2040 -kb 40percentclub/gherkin -km
40percentclub_gherkin_layout_ortho_3x10_mine
```

KB2040 One-Key Keyboard with OLED Display



The KB2040 sports a Stemma QT connector (https://adafru.it/10dD), which is handy for connecting OLED displays. This is supported by QMK, though to make it even easier the author of this guide also contributed a pull request to provide a ready-to-use example with the KB2040 (https://adafru.it/10dE).



Connect pins D3 and D4 to diagonally opposite pins of the tactile switch
Use a cable to connect the STEMMA QT connectors of the KB2040 and OLED display.

You can follow the instructions on GitHub to fetch the pull request to your computer and build it for yourself, or just use the pre-built version below, it's ready to be copied to the RPI-RP2 bootloader drive of a KB2040:

QMK OLED Demo for KB2040

https://adafru.it/10dF

The build steps similar to for the other firmware we've followed in this guide. The keyboard is **handwired/onekey/kb2040** (note it is now **kb2040** and not **rp2040** to indicate the use of the kb2040's Stemma QT connector) and the keymap is **oled**.

These steps only work when you fetched the pull request from github using the commands shown below. It's easiest to just grab the uf2 above.

```
cd ~/qmk_firmware
git fetch origin refs/pull/17786/merge
git checkout FETCH_HEAD
qmk flash -kb handwired/onekey/kb2040 -km oled
```

In either case, you should get a build log displayed in the terminal, and at the end a .uf2 file is created:

```
Ψ Compiling keymap with gmake --jobs=19 --output-sync=target handwired/onekey/
kb2040:oled:flash


QMK Firmware 0.17.5
Making handwired/onekey/kb2040 with keymap oled and target flash

arm-none-eabi-gcc (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.2.1 20201103
(release)
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Generating: .build/obj_handwired_onekey_kb2040/src/
layouts.h                                          [OK]
[Many lines deleted]
Creating binary load file for flashing: .build/
handwired_onekey_kb2040_oled.bin                      [OK]

Size after:
   text    data     bss     dec     hex filename
      0   38284       0   38284    958c handwired_onekey_kb2040_oled.uf2

Creating UF2 file for deployment: .build/
handwired_onekey_kb2040_oled.uf2                           [OK]
Copying handwired_onekey_kb2040_oled.uf2 to qmk_firmware
folder                                       [OK]

Flashing /media/jepler/RPI-RP2 (RPI-RP2)
Wrote 64512 bytes to /media/jepler/RPI-RP2/NEW.UF2
```

Operating the demo

Rather than working as a keyboard, the one button is used to control the demo.

Click the button a short time to switch the mode.

Hold down the button to rotate the display by 90 degree increments.

After a short time, the OLED display will be set to all-black to prevent burn-in.

STEMMA QT Configuration in QMK

The following settings are needed in the appropriate **config.h** file to use the I2C port on the STEMMA QT connector:

```
#define I2C_DRIVER I2CD1
#define I2C1_SDA_PIN GP12
#define I2C1_SCL_PIN GP13
```

The following settings are needed in the appropriate **mcuconf.h** file:

```
#pragma once

#include_next <mcuconf.h>

#undef RP_I2C_USE_I2C0
#define RP_I2C_USE_I2C0 TRUE

#undef RP_I2C_USE_I2C1
#define RP_I2C_USE_I2C1 FALSE
```

If your board has a different pinout, first determine the GP or GPIO number of the pins, and use that instead of GP12/13 in **config.h** for `SDA_PIN` and `SCL_PIN`. Then, determine whether they are on I2C peripheral 0 or 1 according to the pinout of the RP2040, and set the appropriate `USE_I2C` line to `TRUE` and the other to `FALSE` in **mcuconf.h**.

QMK numbers the I2C peripherals differently when it comes to defining the `I2C_DRIVER` so add 1 (I2C0 is I2CD1 and I2C1 is I2CD2). This way, you can use any appropriate pair of pins that are capable of I2C on the RP2040.

To enable the I2C driver and the OLED display, you place lines in **rules.mk**:

```
OLED_ENABLE = yes
OLED_DRIVER = SSD1306

OPT_DEFS += -DHAL_USE_I2C=TRUE
```

The function `oled_task_user` contains code to actually draw to the screen. In the oled example, this is in the file **keyboards/handwired/onekey/keymaps/oled/keymap.c**. The code can draw letters or work by individual pixels. For full info, check out the documentation from QMK's website (https://adafru.it/10dw). Here's a snippet of the code that fills the screen with a test pattern made out of ASCII characters:

```
static void test_characters(void) {
    uint8_t cols       = oled_max_chars();
    uint8_t rows       = oled_max_lines();
    bool    invert     = false;
    uint8_t char_index = 0;
    for (uint8_t row = 0; row &lt; rows; ++row) {
        for (uint8_t col = 0; col &lt; cols; ++col) {
            oled_write_char(get_test_char(char_index), invert);
            if (++char_index &gt;= TEST_CHAR_COUNT) {
                char_index = 0;
                invert     = !invert;
            }
        }
    }
}
```