Table of Contents

Overview
• Required Hardware
• New as of April 2019
• Nordic SDK (DK PCA10056)
• Particle nRF52840 based boards
• Other Boards

nRF52840 Bootloader
• Install nrfjprog
• git clone nRF52840 Bootloader
• Flash the Bootloader with nrfjprog
• Compiling latest bootloader from source
• Test!

Bootloader Details
• nRF52840-DK (PCA10056)

Quick-Start
• Sync your fork
• Check your branch
• Prepare Build!
• Initialize submodules & mpy-cross
• Download Nordic SoftDevice files
• Install adafruit-nrfutil

Build & Flash nRF52840
• Build CircuitPython
• Installing With UF2 (Recommended)
• Uploading with DFU (Harder)

Test nRF8240
• Testing the nRF52840-DK (PCA10056)

Build & Flash Particle
• Option 1: Updating the Bootloader with a Segger J-Link and Arduino IDE.
• Option 2: Manually Burning the Bootloader via nrfjprog
• Installing with UF2
• Testing the Particle
Overview

This guide will take you through the process of building/compiling CircuitPython for the nRF52840 bluetooth LE chipset. Note that CircuitPython for nRF52840 isn't even alpha, its pre-alpha! So this is for bleeding-edge development and experimentation!

Expect things to change, break, and not work as planned.

This guide is not for beginners - Experts only! We make many assumptions about software and skills in this guide!

Required Hardware

This guide will work for the following boards:

- nRF52840-DK - don't use the PDK, it has a pre-release silicon that will not work!

New as of April 2019

Adafruit has created CircuitPython.org to be a resource for CircuitPython software for both Adafruit and non-Adafruit boards. .bin and .uf2 files for loading CircuitPython will be available for compatible boards on that site.

If you only need the .bin or the .uf2 for a particular version of CircuitPython, consider looking on CircuitPython.org.
Nordic SDK (DK PCA10056)

You can click the green button below to go to the page for the Nordic nRF52840-DK (PCA10056).

Go to CircuitPython.org for CircuitPython on the Nordic PCA10056 Development Board

Particle nRF52840 based boards

Go to https://circuitpython.org/downloads() to look for support for Particle boards such as the Boron, Argon, and Xenon.

Other Boards

Scroll through https://circuitpython.org/downloads() to see new boards that might be based on the nRF52840.

To be listed, the developer of the board (or someone else) must have created the correct pin definition files which would allow CircuitPython to know what resources are available on the board. A pull request should be made to the CircuitPython GitHub repo to have support considered for addition. No board will be listed on CircuitPython.org without the appropriate pull request accepted by CircuitPython.org.

nRF52840 Bootloader

Unless you purchased an nRF52840 board from Adafruit, you probably don’t have the right bootloader installed. If you have a PCA10056 (Nordic nRF52840 DevKit) it comes blank, for example

However, with a JLink its super easy to install the bootloader. On the PCA10056 the JLink is even built in!
For this step, plug your computer's USB cable into the MicroUSB port right above the battery.

**Disable Mass Storage on PCA10056 J-Link**

The J-Link firmware on the PCA10056 implement USB Mass Storage, **but this causes a known conflict with reliable USB CDC serial port communication**. In order to use the serial bootloader, you must disable MSD support on the Segger J-Link!

To disable mass storage support, run the J-Link commander, `JLink` or `JLinkExe` (or equivalent) command

and send `MSDDisable`. (You can re-enable MSD support via `MSDEnable`):
Then unplug-replug the dev board. The JLINK disk drive will no longer appear. If you ever want, you can re-enable it later.

Before:

After:

Install nrfjprog

You will almost certainly need to run nordic's nrfjprog tool to burn bootloaders or code

You will first need to install the nrfjprog tool from Nordic Semiconductors for your operating system. The binary files can be downloaded via the following page: https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF5-Command-Line-Tools ()
You will then need to add the `nrfjprog` folder to your system `PATH` variable so that it is available from the command line. The exact process for this is OS specific, but on a POSIX type system like OS X or Linux, you can temporarily add the location to your `PATH` environment variables as follows:

```
$ export PATH=$PATH:YOURPATHHERE/nRF5x-Command-Line-Tools_9_7_2_OSX/nrfjprog/
```

But it really depends on your shell and OS. Basically just make sure you can run `nrfjprog` on the command line. You can test this by running the following command:

```
$ nrfjprog --version
nrfjprog version: 9.7.2
JLinkARM.dll version: 6.20f
```

You may have slightly different version numbers above.

**git clone nRF52840 Bootloader**

```
git clone https://github.com/adafruit/Adafruit_nRF52_Bootloader ()
```

into a handy directory, then open up a terminal in that location and run

```
git submodule update --init --recursive
```

©Adafruit Industries
Flash the Bootloader with **nrfjprog**

This operation only needs to be done once, and only on boards that don't already have the serial bootloader installed.

**nrfjprog** is Nordic's helper program that will talk to JLink devices - the DK has a JLink built into it. You could also use these commands with an everyday JLink and a custom board with SWD port.

Erase chip

Once **nrfjprog** is installed and available in PATH you can test nrfjprog and also erase your chip with:

```
nrfjprog -f nrf52 --eraseall
```

If you get an error that there is no debugger connected to the PC, check you're plugged into the right port, and have JLink software/drivers installed.

Compiling latest bootloader from source

You can then burn the version you see with a line that matches the subdirectory (e.g. pca10056 or feather_nrf52840_express) and then the version number:

```
make BOARD=pca10056 clean
make BOARD=pca10056 sd
make BOARD=pca10056 flash
```
Test!

Plug a USB cable into the nRF52 USB microUSB port (not the JLink port) and you should see a new disk drive appear:

It contains a INFO_UF2.txt for reading the details of the bootloader.
If you can't flash directly from the 'make' command, you can create a bootloader hex file with the 'genhex' flag.

```
maker BOARD=pca10056 genhex
```

Then you can directly burn the hex file with:

```
nrfjprog -f nrf52 --program pca10056_bootloader.hex
```

---

**Bootloader Details**

Unlike the SAMD21 and SAMD51, you cannot 'abuse' the reset button to have a double-click-reset for jumping into the bootloader.

Instead, we have a separate button for entering bootloader or factory reset mode

**nRF52840-DK (PCA10056)**

Holding down Button1 while clicking RESET will put the board into USB bootloader mode

Holding down both Button1 & Button2 while clicking RESET will put the board into OTA (over-the-air) bootloader mode

---

**Quick-Start**

If you don't want to compile CircuitPython you can just load the most recent build using UF2 with drag-n-drop.

Click here to find the latest build:
Click the directory for your board...

- feather m0 express circuitpy
- feather m0 rfm69/
- feather m0 rfm9x/
- feather m4 express/
- feather nrf52832/
- feather nrf52840 express/
- feather radiofruit zigbee/
- gemma m0/
- hallowing m0 express/
- itsybitsy m0/
- itsybitsy m0 express/
- itsybitsy m4 express/
- metro m0 express/
- metro m4 express/
- metro m4 express revb/
- pca10056/
- pirkey m0/
- trinket m0/

By default the newest build is at the top. Check the two letter code for your language: english, espagnol, french, filipino (tagalog) - we'll be using the english build but instructions are the same for other languages.

Enter the nRF52840 bootloader by double-clicking if it isn't already in bootloader mode, then drag the UF2 file to NRFBOOT

You'll see a new disk drive CIRCUITPY
Prepare to Build

If you want to compile your own build of CircuitPython, follow these steps instead

If you followed the Quick Start page and just want to load the latest build, you can skip this page and the next one as well and go straight to Testing!

We have an excellent guide that we maintain with details on how to build CircuitPython here:

https://learn.adafruit.com/building-circuitpython

There are a few differences for the CircuitPython build of nRF52 since we have to fetch special files from Nordic!

Sync your fork

nRF52 support is changing almost daily. If you are using the adafruit repo as your git repository dont forget to do a `git pull`.

If and only if you're working on a fork don't forget to

```
git pull --rebase upstream master
git push --force-with-lease origin master
```

Check your branch

There are various branches of work - the latest version may not be in `master`.

Prepare Build!

Here's things you only have to do once
Initialize submodules & mpy-cross

As documented here () start with

```
cd circuitpython
git submodule sync
git submodule update --init --recursive
make -C mpy-cross
```

Download Nordic SoftDevice files

You then need to download the soft devices from Nordic, download the files by running:

```
  cd ports/nrf
  ./bluetooth/download_ble_stack.sh
```

This script relies on wget, which must be available from the command line.

You should see successful downloads and when done, you can list the *softdevice*hex's you downloaded - there may be a few different version numbers.
Install adafruit-nrfutil

OK now you're going to install our helper to program the firmware over the bootloader. Note that above we installed the software for burning the bootloader, but you'll only want to do that once and then just rely on the bootloader.

adafruit-nrfutil is on PyPI, and you can install it on the command line with

```
pip3 install --upgrade adafruit-nrfutil
```

nrfutil is requires Python 3 - So use pip3 and/or set up your python environment for v3

Verify you installed 0.5.3.post9 or later

Build & Flash nRF52840

Build CircuitPython

Now we're back to following this guide ()
Go back to your circuitpython repository. From within `ports/nrf` build the circuitpython runtime with:

```
make BOARD=pca10056
```

Upon success you should see that a `build-boardname-softdeviceversion/firmware.hex` and `firmware.uf2` file has been built!

### Installing With UF2 (Recommended)

UF2 is a drag-n-drop bootloader system and it's the easiest way to install. Locate the firmware.uf2 file in the build directory. Then drag it onto NRF52BOOT disk drive.

It will disappear and then a new drive will appear named CIRCUITPY - this is the internal circuitpython storage.

---

On the PCA10056, you can re-start the bootloader by holding down Button 1 while clicking the reset button.
Uploading with DFU (Harder)

If for some reason you don't want to use UF2, you can also use DFU which is the standard method for programming microcontrollers, it requires special software tho so we recommend UF2.

Once the firmware is built as a hex we have to package it up for the bootloader in a zip file. Do this by running:

```
make BOARD=pca10056 dfu-gen
```

You should see the creation of a `build-boardname-softdeviceversion/dfu-package.zip` file.

Now all we have to do is upload it. Select the serial port that you find appears when the bootloader is running and put that next to `SERIAL=`

```
make BOARD=pca10056 SERIAL=/dev/ttyS2 dfu-flash
```

Or if you have a specific zip you want to upload:

```
adafruit-nrfutil --verbose dfu serial --package build-pca10056-s140/dfu-package.zip -p COM3 -b 115200 --singlebank
```

© Adafruit Industries
Test nRF8240

Now you can connect with Mu or your favorite serial port software to the CircuitPython runtime, if you hit Control-C a few times you'll get this notice of the build version!

Testing the nRF52840-DK (PCA10056)

Save the following to `code.py` on the disk drive and re-load it by typing Control-D in the REPL to see LED1 blink!

```
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led = DigitalInOut(board.P0_13)
led.direction = Direction.OUTPUT

while True:
    led.value = False
    time.sleep(0.1)
    led.value = True
    time.sleep(0.1)
```

You can also read the four buttons and light up corresponding four LEDs on the DK like so:

```
# CircuitPython I0 demo #1 - General Purpose I/O
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led_pins = [board.P0_13, board.P0_14, board.P0_15, board.P0_16]
button_pins = [board.P0_11, board.P0_12, board.P0_24, board.P0_25]
```
led_outputs = []
for led_pin in led_pins:
    led = DigitalInOut(led_pin)
    led.direction = Direction.OUTPUT
    led_outputs.append(led)

button_inputs = []
for button_pin in button_pins:
    button = DigitalInOut(button_pin)
    button.direction = Direction.INPUT
    button.pull = Pull.UP
    button_inputs.append(button)

while True:
    for i, button in enumerate(button_inputs):
        if not button.value:
            print("Button #", i+1, "pressed!")
            led_outputs[i].value = button.value
    time.sleep(0.01)

Build & Flash Particle

Support for the Particle Argon () and Xenon () boards featuring the nRF52840 is also being built into CircuitPython. The process at present requires an advanced J-Link programmer and a number of steps, so at present it is not recommended for beginners.

Option 1: Updating the Bootloader with a Segger J-Link and Arduino IDE.

You can burn the bootloader from within the Arduino IDE using a Segger J-Link. We've detailed these steps on this Learn Guide ()

Note: When you’re ready to burn the bootloader from the Arduino IDE, make sure the following options are selected:

- Select `Tools > Board > Adafruit Bluefruit Feather52840 Express`
- Select `Tools > Programmer > J-Link for Bluefruit nRF52`
- Select `Tools > Burn Bootloader` with the board and J-Link connected
Option 2: Manually Burning the Bootloader via nrfjprog

The next option is to manually burn the bootloader from the command line using nrfjprog from Nordic.

Note: These steps will assume you're using a Particle Argon board, however, the Adafruit Bluefruit nRF52 Bootloader can be used with the following Particle boards - Xenon, Boron, and Argon. These boards are supported by CircuitPython 4 Alpha 5 and up.

The Particle Argon has both the nRF52840 and a ESP32 onboard. Before you can use the ESP32 as a coprocessor in CircuitPython, you'll need to replace the Particle's firmware with CircuitPython.

To cleanup the directory, we'll first run:

```
make BOARD=particle_argon clean
```

We'll build the CircuitPython runtime, by running:

```
made board=particle_argon
```

The next few commands will use Nordic's nrfjprog tool to burn the bootloader of the Particle Argon with the BlueFruit nRF52-Bootloader.

```
By burning the Adafruit nRF52-Bootloader on the Particle, you are overwriting the firmware already on your Particle board.
```

Make sure your Particle is plugged into a debugger, and that both the debugger and the Particle are connected to the PC.

We'll start by erasing all of the flash by running:

```
make BOARD=particle_argon erase
```

Then, we'll flash Nordic SoftDevice (and chip erase) by running:
Finally, we'll flash the Bluefruit NRF52 bootloader onto the Particle by running:

```
make BOARD=particle_argon flash
```

If everything went well, you'll notice a new drive named ARGONBOOT (or XENONBOOT, if you're following along with a Particle Xenon). If you used Option 1 (Installing via Arduino IDE), you'll see a drive called FTHR840BOOT.

Next, and we'll install the CircuitPython firmware for the Particle board you're using. Head to the CircuitPython release page and search for the board you're using.
Note: Support for the Particle Argon, Xenon, and Boron were first added in CircuitPython 4.0.0 Alpha 5.

Once downloaded, locate the adafruit-circuitpython-particle_argon-X.X.X.uf2 that you just downloaded. Then, drag it onto the ARGONBOOT drive.

It will disappear and then a new drive will appear named CIRCUITPY - this is the internal circuitpython storage.

Testing the Particle

Since your Particle board is running CircuitPython, you can connect with Mu () or your favorite serial port software (like Screen or PuTTY) to the CircuitPython runtime.

If you hit Control-C a few times you'll get this notice of the build version

There's a bright-blue LED built into the Particle and we're going to make it blink. Save the following as code.py on the drive and reload it by typing Control+D into the REPL to see BLUE_LED blink!

```python
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led = DigitalInOut(board.BLUE_LED)
led.direction = Direction.OUTPUT

while True:
    led.value = False
    time.sleep(0.1)
```
led.value = True
time.sleep(0.1)

There's an unused MODE button (the reset button is used by CircuitPython to reset the board). We can test that the MODE button works by using it to control the blue LED.

```python
# CircuitPython on Particle Demo - GPIO
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led_pin = DigitalInOut(board.BLUE_LED)
led_pin.direction = Direction.OUTPUT
led_pin.value = False

button = DigitalInOut(board.MODE)
button.direction = Direction.INPUT

while True:
    # Check for a button press
    if not button.value:
        print('Button Pressed!')
        led_pin.value = button.value
    time.sleep(0.01)
```