



SASC Physics 2020

Rotary Macropad



QMK and Key Map Guide

This guide makes the following assumptions:

- A. A basic understanding of how to download from a Github repository via web browser
- B. A build environment is available to you and you are able to navigate this environment.
- C. Use of a software package for editing the QMK files such as VSCode, Notepad ++, appropriate for C type languages.
- D. QMK Toolbox is installed
- E. Appropriate USB libraries are installed.
- F. The Macropad is build and ready for flashing
- G. The macropad is built with an encoder in the top left position as per build instructions.
- H. A data-capable Micro-USB cable is on hand.

The Completed Newbs Guide to QMK can assist with many of the points above.
(<https://docs.qmk.fm/#/newbs?id=the-complete-newbs-guide-to-qmk>)

Visual Studio Code (VSCode) - <https://code.visualstudio.com/>
Notepad++ - <https://notepad-plus-plus.org/>

QMK Toolbox - https://github.com/qmk/qmk_toolbox/releases

It is also recommended that you gain some familiarity with the QMK Key Codes
<https://docs.qmk.fm/#/keycodes>

This guide aims to help understand the minimum editing of the QMK files required to re-map the keymaps of the Rotary Macropad that you have, or are about to build.

By the end of this guide, you should have an understanding of:

- i. What to edit
- ii. Where to edit
- iii. How to produce a .hex file for flashing
- iv. How to flash your macropad

1. Retrieving the QMK keyboard Files.

- Navigate to the repository on github at: <https://github.com/flehrad/DRMP>
- Click on the Green "Code" box above the repository contents, and click on the "Download ZIP" (Figure 1)

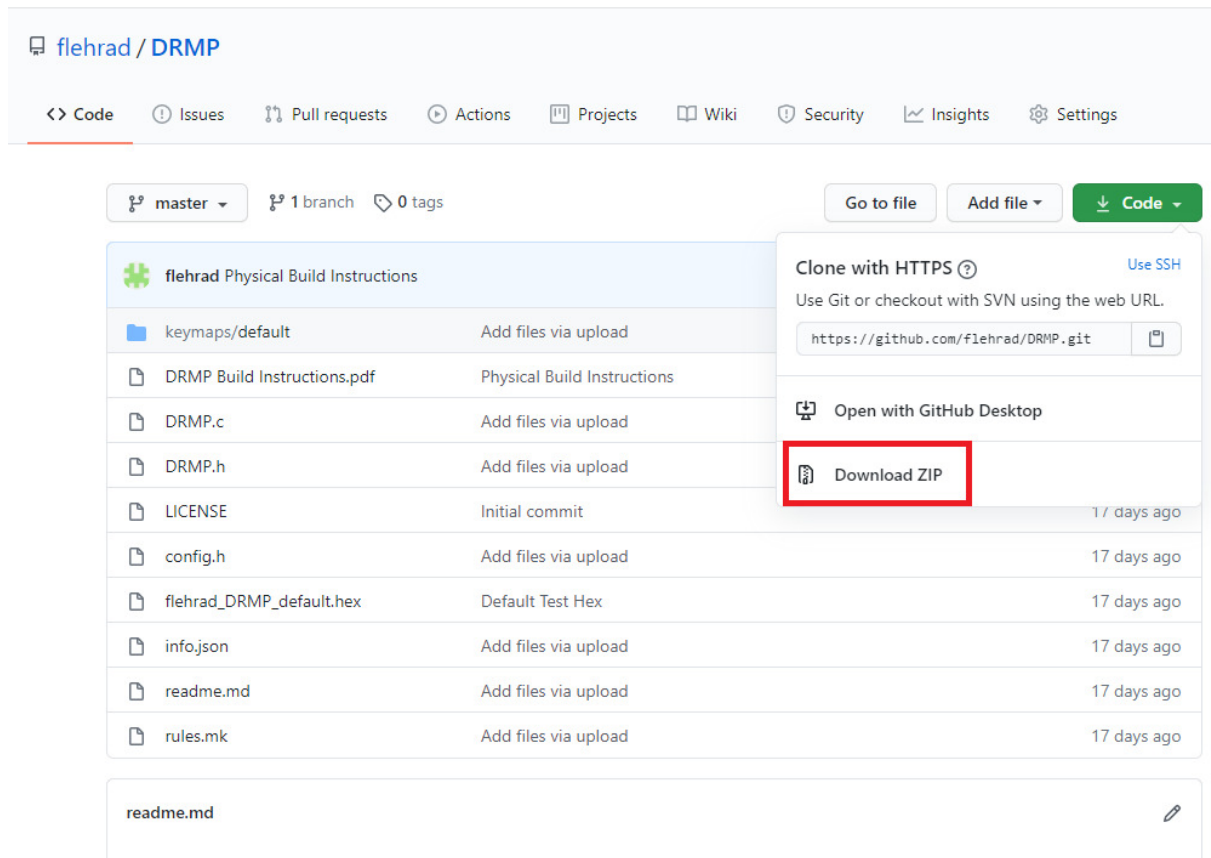


Figure 1. Github download of QMK keymap files.

- Save the zip file somewhere convenient, and unpack the contents into your QMK firmware folder in a location you are able to identify and find. Figure 2 shows the environment that this firmware was built in. The github repository will contain more files than Figure 2 shows.

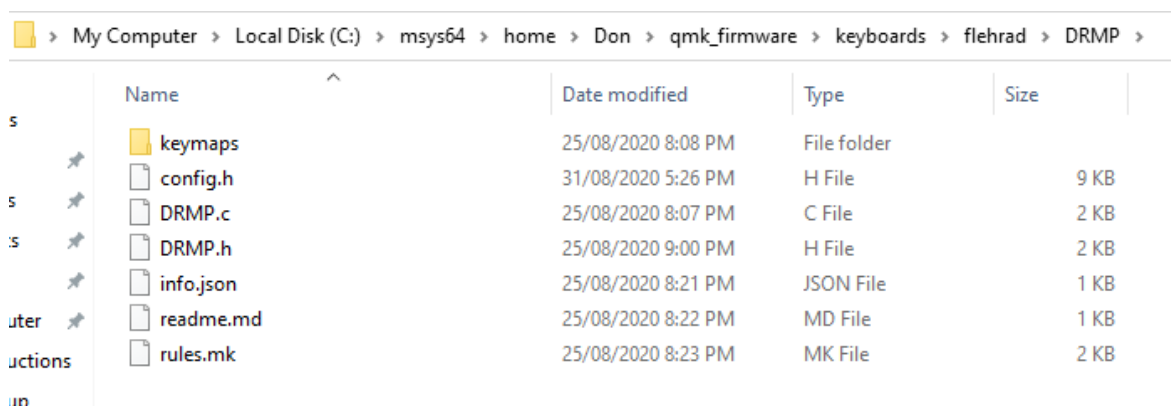


Figure 2. QMK firmware folder example.

The two files you will may be editing are the config.h file in the root directory, and the keymap.c file inside the keymaps folder.

2. Editing the QMK keyboard files.

Reference figures here are from code opened in VSCode

- Open your editor and navigate to open the keymap.c file inside the keymaps folder as per 1.c.
- Inside keymap.c, there will be 87 lines of code, which contains the information for the keymap of the macropad, any macros, and the rotary encoder outputs.
- In the section from lines 29 to 36 is the default matrix of the macropad as below:

```
28
29  const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
30      /* Base */
31      [_BASE] = LAYOUT([
32          KC_1,  KC_2,  KC_3,
33          KC_4,  KC_5,  KC_6,
34          KC_7,  KC_8,  KC_9
35      ]),
36  };
37
```

Figure 3. Keymap Matrix

- The grid of 3 x 3 positions are marked with the keycodes KC_1 through to KC_9 and correspond to the physical layout of the macropad when the encoder is built in the top left position.
- Each position is separated by a comma (,) except for the last position which is closed and comma delimited on line 35. When editing, do not delete any of the commas or brackets.

2.1. Keycode Output

- Edit the matrix accordingly with KC_ codes from the QMK key codes lists. Note that the rotary encoder in KC_1 position has a push-button switch output that is represented by the KC_1 position.
- Save the file when editing is completed.

2.2. Macro Output

- It is possible to output Macros in QMK. This code occurs in three components.

- The first component is defining custom_keycodes starting from line 24

```
23  // Defines the keycodes used by our macros in process_record_user
24  enum custom_keycodes {
25      QMKBEST = SAFE_RANGE,
26      QMKURL
```

Figure 4. Macro names defined.

- The second section is the matrix as above in section 2.c, and edited as per 2.1.a

- iii. The third section is where the macro is specifically defined, when the custom_keycode is pressed in the matrix position, in lines 38 through to 58.

```
37
38 bool process_record_user(uint16_t keycode, keyrecord_t *record) {
39     switch (keycode) {
40         case QMKBEST:
41             if (record->event.pressed) {
42                 // when keycode QMKBEST is pressed
43                 SEND_STRING("QMK is the best thing ever!");
44             } else {
45                 // when keycode QMKBEST is released
46             }
47             break;
48         case QMKURL:
49             if (record->event.pressed) {
50                 // when keycode QMKURL is pressed
51                 SEND_STRING("https://qmk.fm/\n");
52             } else {
53                 // when keycode QMKURL is released
54             }
55             break;
56     }
57     return true;
58 }
```

Figure 5. Macro recording section.

- b. First define the name of the Macro. In the sections above, QMKBEST and QMKURL are already defined as per Figure 4.
- c. Insert the Macro name to the corresponding matrix position that will be triggered by the keypress. In Figure 5, QMKBEST is assigned to KC_1 position and QMKURL to the KC_9 position.

```
29 const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
30     /* Base */
31     [_BASE] = LAYOUT(
32         QMKBEST, KC_2, KC_3,
33         KC_4, KC_5, KC_6,
34         KC_7, KC_8, QMKURL
35     ),
36 };
```

Figure 6. Macro names inserted into keymap

- d. When the switch pressed is for the QMKBEST or QMKURL positions, it will output the string as defined in the recorded section, i.e. for QMKBEST, the output in line 43 in Figure 5 and for QMKURL, the output in line 51 in Figure 5.
- e. For non-string macros such as using modifiers (Ctrl, Alt, etc.), more detailed explanation with example can be found at https://docs.qmk.fm/#/feature_macros

2.3 Rotary Encoder Outputs

- a. In lines 73 to 87 is the code block for the output of the rotary encoder. The default code accounts for 2 rotary encoders, however only the first encoder will be used by this macropad (Figure 7).

```
73 void encoder_update_user(uint8_t index, bool clockwise) {
74     if (index == 0) { /* First encoder */
75         if (clockwise) {
76             tap_code(KC_R);
77         } else {
78             tap_code(KC_L);
79         }
80     } else if (index == 1) { /* Second encoder */
81         if (clockwise) {
82             tap_code(KC_DOWN);
83         } else {
84             tap_code(KC_UP);
85         }
86     }
87 }
```

Figure 7. Rotary Encoder code block

- b. While Line 75 defines the output for what it calls “clockwise” orientation, this is dependent upon how the EC-11 footprint was utilised during the design of the PCB itself. It is better to think of the word ‘clockwise’ here as simply a label of one direction of output.
- c. The output for rotation in the labelled direction ‘clockwise’ is in the tap_code output in line 76, in this case, the output is KC_R.
- d. The other direction to the Clockwise label is then output as line 78, in this case KC_L.
- e. If the rotation of the encoder produces outputs in the incorrect direction, an additional edit is required to correct this.
- Open the file config.h from the root directory of this keyboard set, not the config.h from inside the keymaps/default folder.
 - Line 255 is by default enabled for Top Left encoder orientation, however, if a reversal is required, simply add two “//” to the front of the code to deactivate it as a comment (Figure 8).

a.

b.

Figure 8a. Active Encoder direction flip; b. Deactivated Encoder direction flip

- f. Save files to retain the changes.

3. Making the .hex file

- a. Enter your build environment
- b. Navigate through to qmk_firmware folder using "cd" commands, e.g. "cd qmk_firmware" (Figure 9).

```
Don@Don-Win10 MINGW32 ~  
$ cd qmk_firmware/  
  
Don@Don-Win10 MINGW32 ~/qmk_firmware
```

Figure 9. Navigating to QMK_firmware

- c. Use the 'make' command with your directory structure and the default keymap to start the compiling of the .hex file (Figure 10).

```
Don@Don-Win10 MINGW32 ~/qmk_firmware  
$ make flehrad/DRMP
```

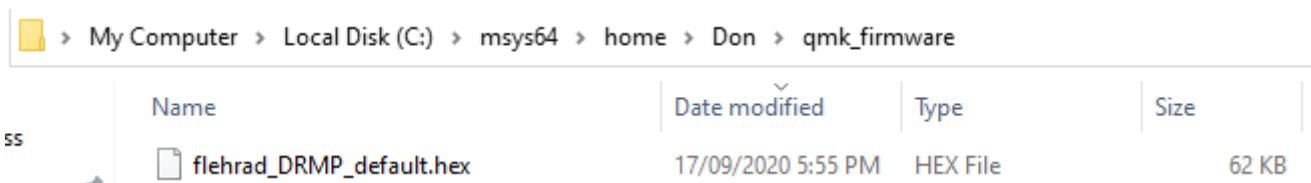
Figure 10. Make command

- d. The output will produce a number of compile lines with [OK] if there are no code errors, and a line output indicating the .hex file is ready for use (Figure 11).

```
Compiling: lib/lufa/LUFA/Drivers/USB/Core/Events.c  
[OK]  
Compiling: lib/lufa/LUFA/Drivers/USB/Core/HostStandardReq.c  
[OK]  
Compiling: lib/lufa/LUFA/Drivers/USB/Core/USBTask.c  
[OK]  
Linking: .build/flehrad_DRMP_default.elf  
[OK]  
Creating load file for flashing: .build/flehrad_DRMP_default.hex  
[OK]  
Copying flehrad_DRMP_default.hex to qmk_firmware folder  
[OK]  
Checking file size of flehrad_DRMP_default.hex  
[OK]  
* The firmware size is fine - 22506/28672 (78%, 6166 bytes free)  
  
Don@Don-Win10 MINGW32 ~/qmk_firmware  
$
```

Figure 11. The .hex file is complete.

- e. The .hex file will now be in the qmk_firmware root directory ready for use (Figure 12).



Name	Date modified	Type	Size
flehrad_DRMP_default.hex	17/09/2020 5:55 PM	HEX File	62 KB

Figure 12. The .hex file is ready for use.

4. QMK Toolbox and Flashing

- a. Open QMK Toolbox
- b. Use the 'Open' button and navigate to the .hex file compiled in 3.e.
- c. Ensure the Microcontroller dropdown selection is the 'atmega32u4' option (as this is the integrated circuit chip used on the Pro-Micro microcontroller).
- d. Tick the Auto-Flash option checkbox.
- e. The dialogue box should now look like Figure 13 below.

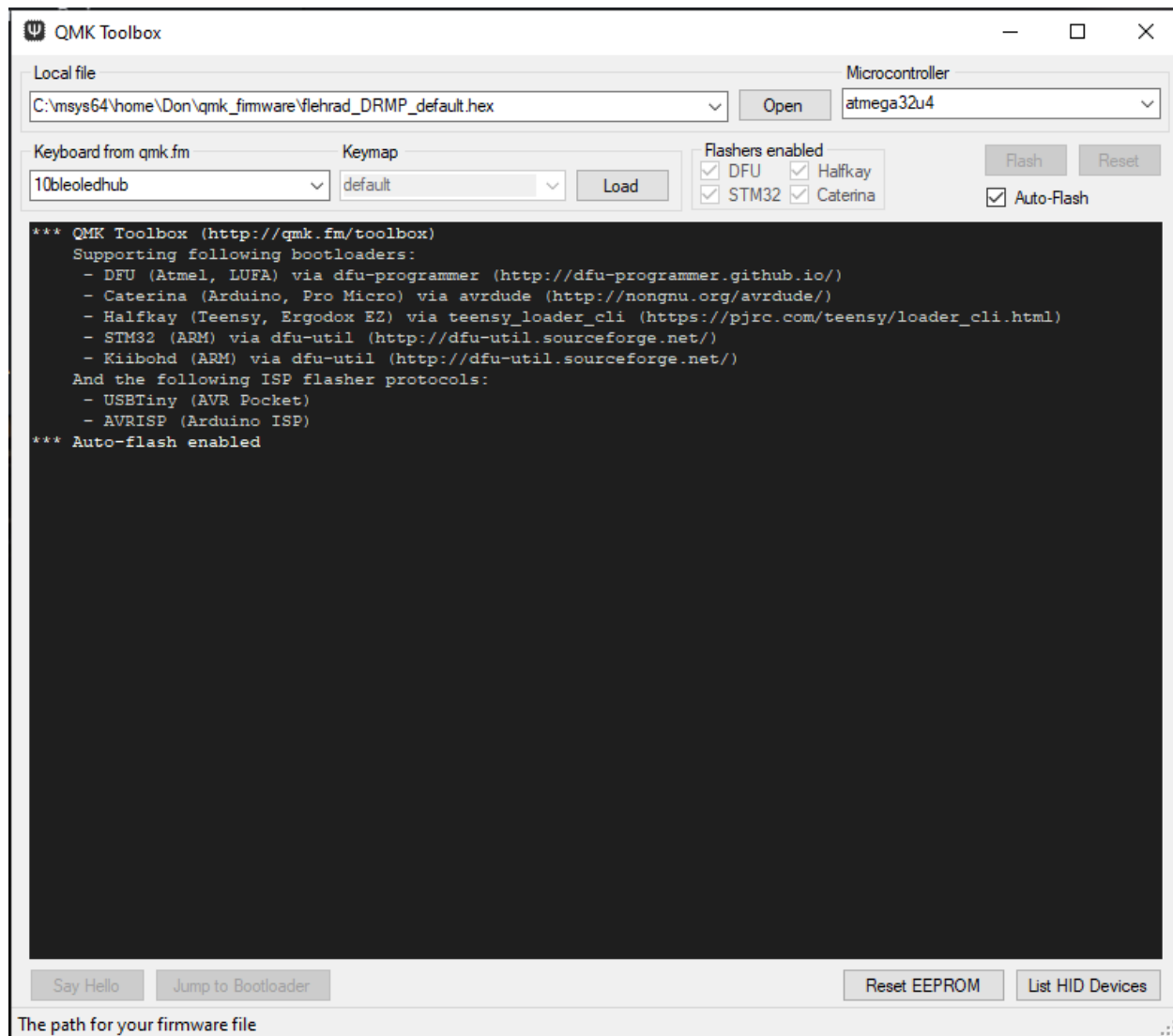


Figure 13. QMK Toolbox prepped for loading

- f. Plug in the macropad using a micro-USB cable to the computer.
- g. Some dialogue will appear indicating the Pro-micro is connected and recognised by the software.

CAUTION: Do not remove the micro-USB cable until the macropad flashing is complete as indicated by QMK Toolbox dialogue. Early removal may brick your Pro-Micro.

- h. Press the reset switch on the macropad and the Pro-micro will be put into bootloader mode for eight (8) seconds.
- i. As the auto-flash checkbox was ticked, QMK Toolbox should automatically recognise the Pro-micro is in bootloader mode and attempt to flash the Pro-micro with the loaded .hex file. Do not unplug the micro-USB cable until the process is complete (Figure 14 and Figure 15).

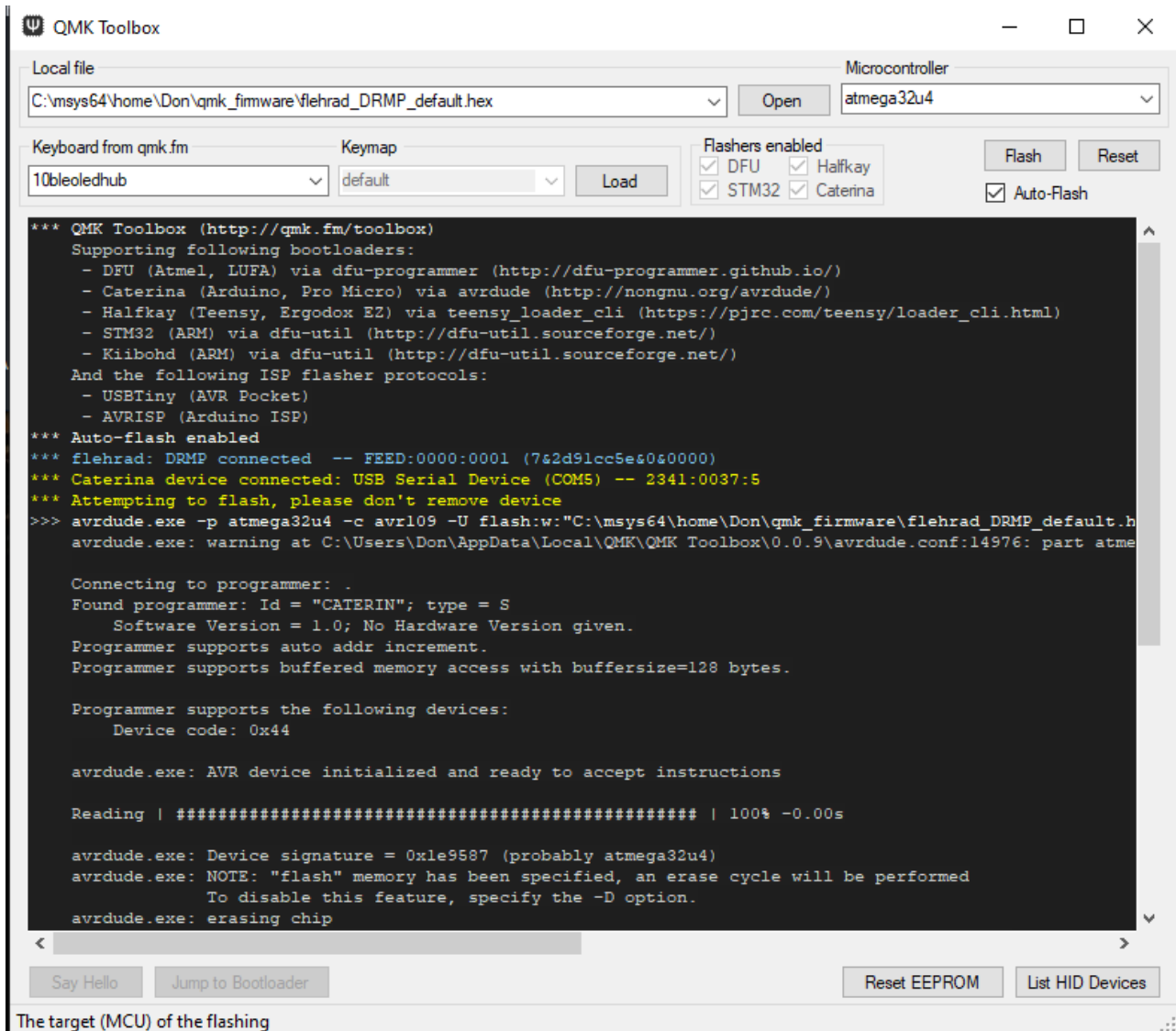


Figure 14. Auto-flashing commencing, note the yellow warning text.

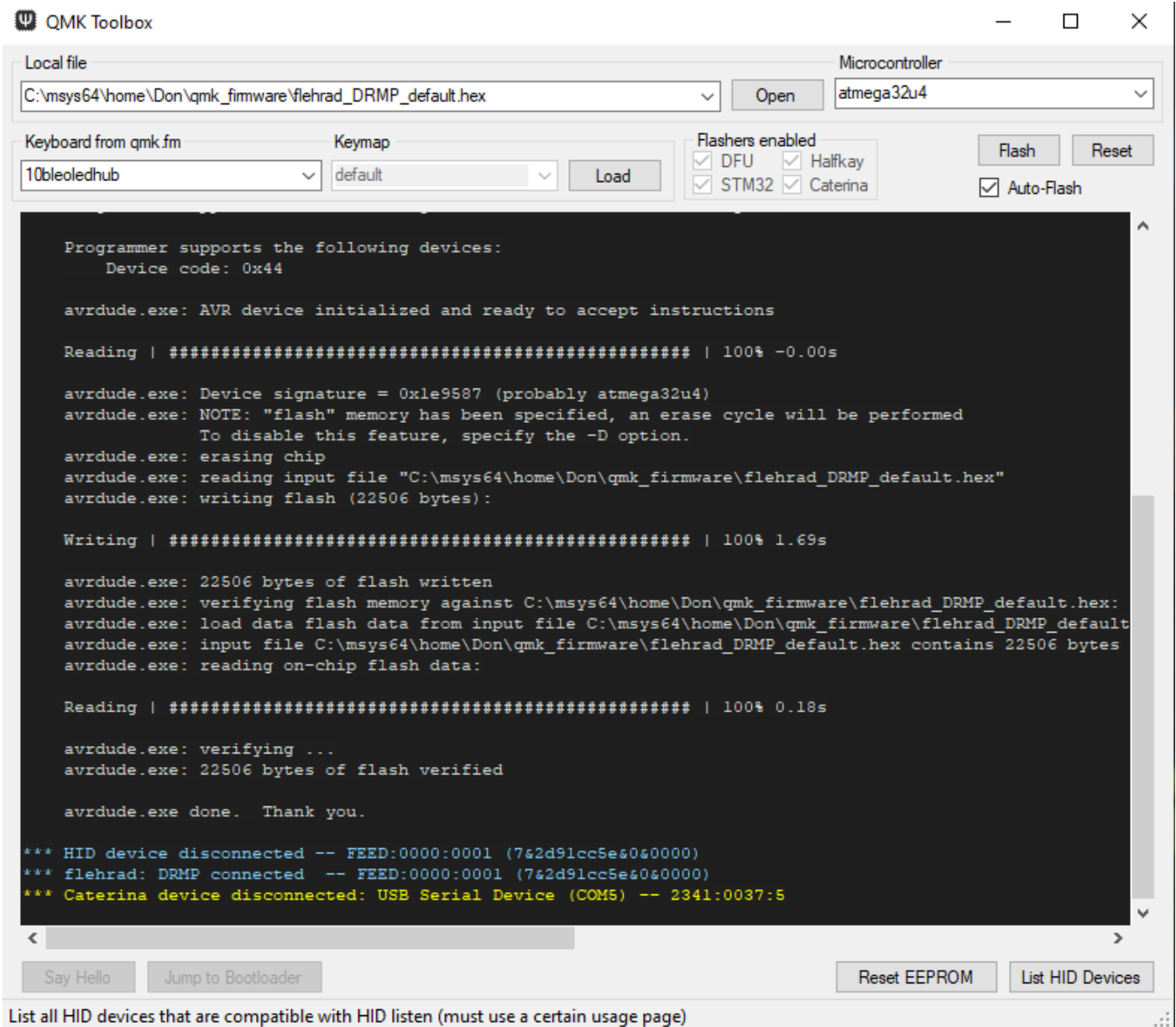


Figure 15. Completion ends with the device being automatically disconnected.

- j. Once a successful flashing of the firmware is complete, QMK Toolbox will automatically disconnect the device.
- k. Unplug the micro-USB cable from the macropad.
- l. Close QMK Toolbox.
- m. Plug in the micro-USB cable to macropad.
- n. Test macropad functions that have been saved in the keymaps, in notepad or media players and so forth.

Congratulations, your rotary macropad is now ready for use.

For troubleshooting any QMK Code, refer to the QMK Website or join the QMK Discord.