QMK
Proton
Drop-in Pro Micro replacement with break-off speaker mount
STM32F303CCT6
USB Type-C
3.3V 72MHz
256KB Flash
48KB SRAM
Made possible by the QMK community

FROM OPEN SOURCE TO OPEN SEASON

====================================

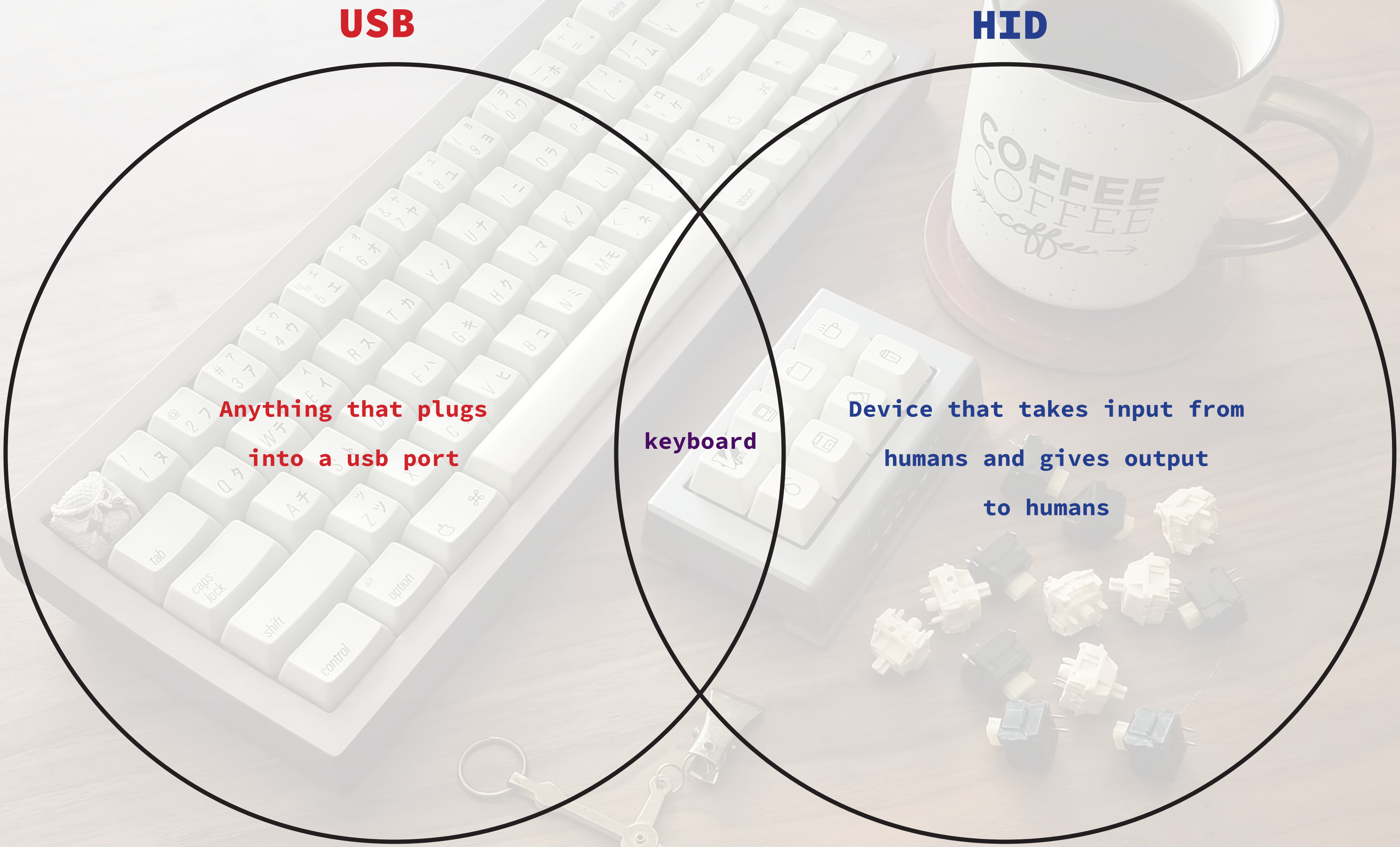
Using and abusing debug features to CTRL+F Your way into a tool(kit)

C.Y.O // TTYlerDurden

OVERVIEW:

- What is USB/HID
- How a system identifies and utilizes USB devices
- How a keyboard procceses keystrokes
- QMK, scancodes/keycodes, modifying keycodes and keyboard shortcuts
- Macropads and MicroControllers
- Flashing QMK
- QMK files structure
- Debugging
- Walkthrough for modifying QMK files and capturing debug messages
- Abusing debugging in real world examples

WHAT IS USB/HID:



USB

HID

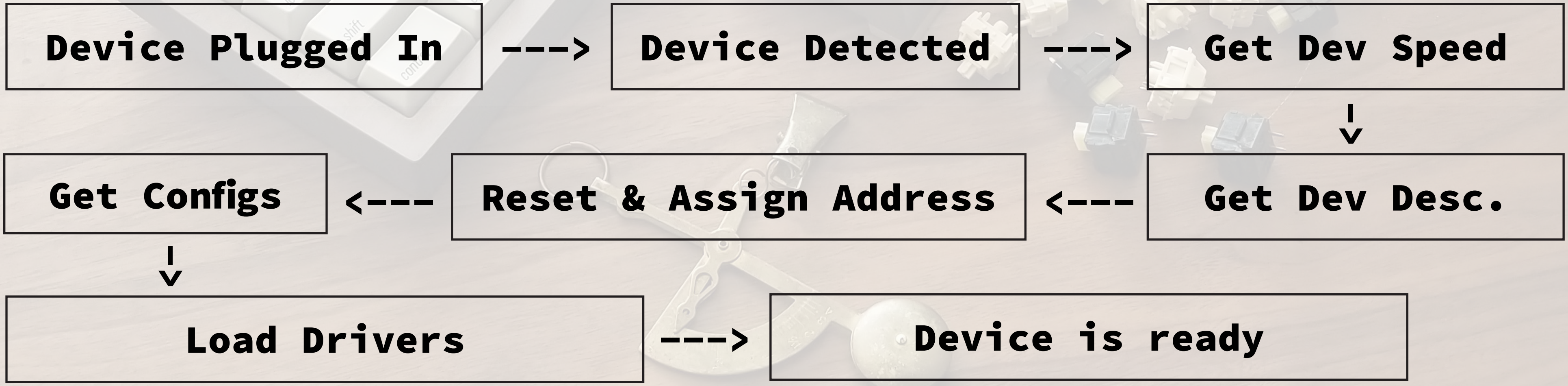
Anything that plugs into a usb port

keyboard

Device that takes input from humans and gives output to humans

DETECTING A USB DEVICE:

- Before you can use a USB device, your system must enumerate the device for info about it.
- The USB bus is designed so that the act of inserting (and removing) devices can be recognized by the host.
- When this happens, the host informs its device driver which scans the bus and asks the device to identify itself.
- These questions of consist calls made by the host to the device for descriptors.
- These descriptors stored in the device describe its capabilities and all the basic information about it including the device type (in our case HID) vendor id, product id, and more.
- The device must respond to these calls (along with any other information it may be sending or receiving) with these decriptors IAW the specifications found in the report descriptor.
- This provided information tells the host system what drivers to use and how to process the information coming in through that USB bus.
- Once successfully enumerated, the host can begin sending and receiving data

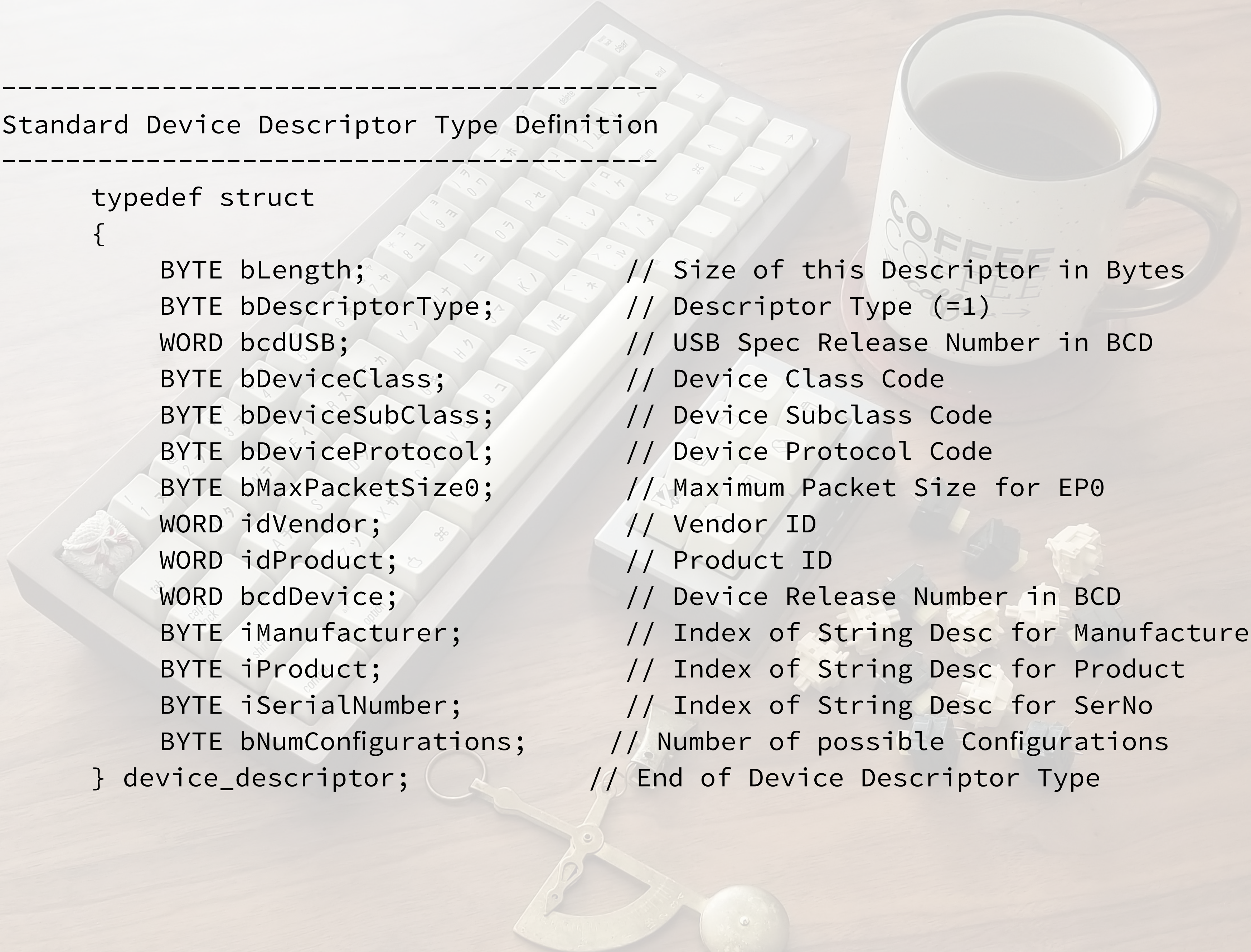


Device Plugged In ---> Device Detected ---> Get Dev Speed

Get Configs <--- Reset & Assign Address <--- Get Dev Desc.

Load Drivers ---> Device is ready

EXAMPLE:



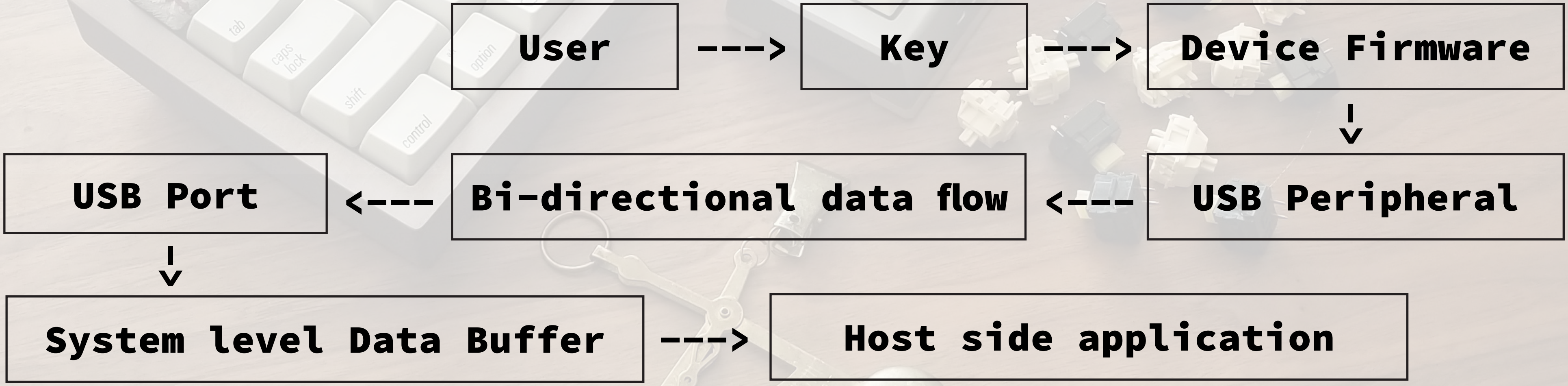
```
//------------------------------------------
// Standard Device Descriptor Type Definition
//------------------------------------------
         typedef struct
         {
             BYTE bLength;                // Size of this Descriptor in Bytes
             BYTE bDescriptorType;        // Descriptor Type (=1)
             WORD bcdUSB;                 // USB Spec Release Number in BCD
             BYTE bDeviceClass;           // Device Class Code
             BYTE bDeviceSubClass;        // Device Subclass Code
             BYTE bDeviceProtocol;        // Device Protocol Code
             BYTE bMaxPacketSize0;        // Maximum Packet Size for EP0
             WORD idVendor;               // Vendor ID
             WORD idProduct;              // Product ID
             WORD bcdDevice;              // Device Release Number in BCD
             BYTE iManufacturer;          // Index of String Desc for Manufacturer
             BYTE iProduct;               // Index of String Desc for Product
             BYTE iSerialNumber;          // Index of String Desc for SerNo
             BYTE bNumConfigurations;    // Number of possible Configurations
         } device_descriptor;           // End of Device Descriptor Type
```

HOW KEYBOARDS REGISTER KEYPRESSES:

- When you press a key, your keyboard is capable or recognizing this as an event (pressed, held, or released)
- Your keyboard then transfers those key presses to the host in the form of a keyboard report containing scancodes, which identifies the current state of the board.
- The firmware does not send actual letters or characters, only scancodes
- Once the keycode (scancode) is sent to the OS, software has to match it to an actual character
- The HID specification defines what a keyboard can actually send through USB and be properly recognised
- This includes a pre-defined list of scancodes which are simple numbers from 0x00 to 0xE7

```
KC_A == 0x04
```

- Each press is detected through a process called Matrix scanning.
- This happens many times a second (approximately 10x per second)
- Essentially a keypress is detected as 1 instead of a 0



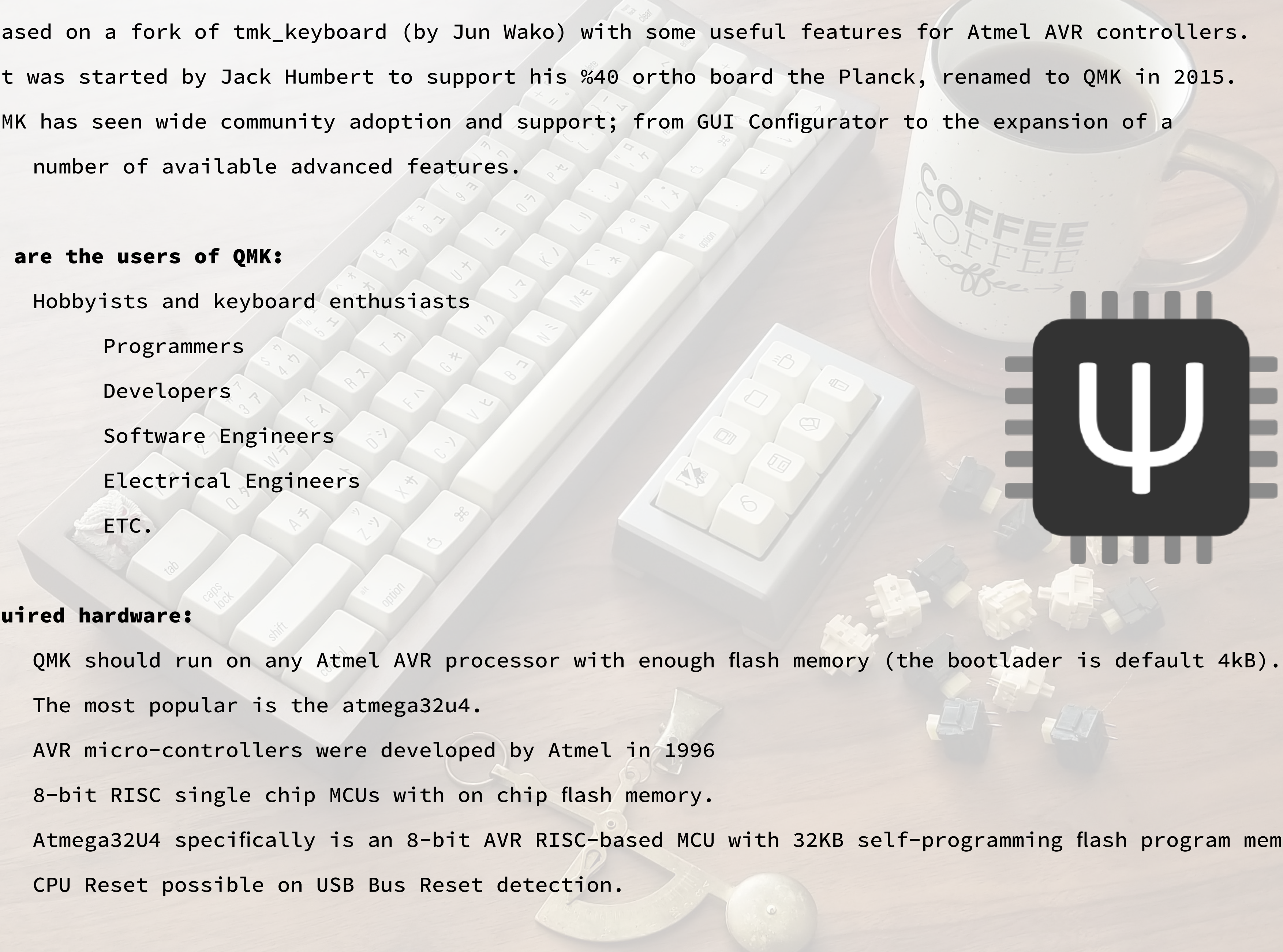
User ---> Key ---> Device Firmware

USB Peripheral

USB Port <--- Bi-directional data flow <--- USB Peripheral

System level Data Buffer ---> Host side application

WHAT IS QMK?



- QMK (Quantum Mechanical Keyboard) is an open source community that maintains QMK Firmware.
- Based on a fork of tmk_keyboard (by Jun Wako) with some useful features for Atmel AVR controllers.
- It was started by Jack Humbert to support his %40 ortho board the Planck, renamed to QMK in 2015.
- QMK has seen wide community adoption and support; from GUI Configurator to the expansion of a number of available advanced features.

Who are the users of QMK:

Hobbyists and keyboard enthusiasts

Programmers

Developers

Software Engineers

Electrical Engineers

ETC.

Required hardware:

QMK should run on any Atmel AVR processor with enough flash memory (the bootlader is default 4kB).

The most popular is the atmega32u4.

AVR micro-controllers were developed by Atmel in 1996

8-bit RISC single chip MCUs with on chip flash memory.

Atmega32U4 specifically is an 8-bit AVR RISC-based MCU with 32KB self-programming flash program memory

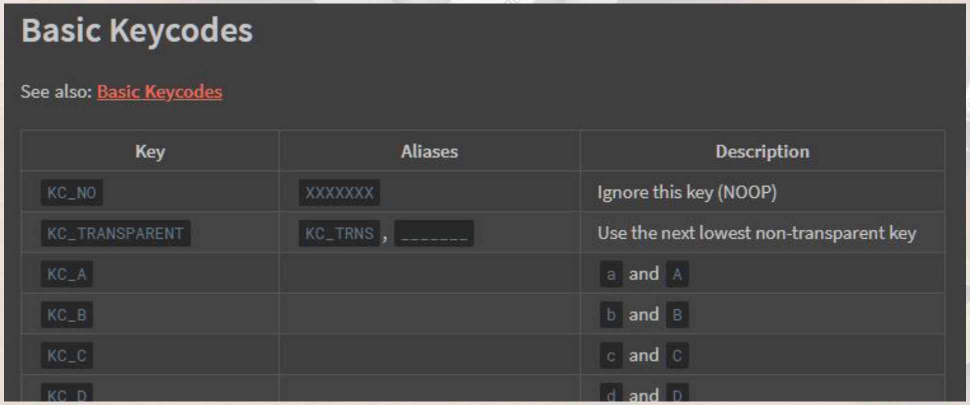
CPU Reset possible on USB Bus Reset detection.

SCANCODES:

- Your keyboard identifies the state of every switch and maps it to a keycode (scancode) via the help of a C macro, or keyboard layout.
- Matrix scanning cares about changes since last scan.

  1 = pressed / 0 = not pressed
- QMK stores the last scan, and if different than the current scan, it detects which key was pressed.
- <keyboardname>.h -> keymap.c

Physical switch location -> matrix -> logical switch location -> user keymap (which specifies the keycode, or any other user defined variable, for the logical switch location)



Basic Keycodes

See also: Basic Keycodes

Key	Aliases	Description
KC_NO	XXXXXXX	Ignore this key (NOOP)
KC_TRANSPARENT	KC_TRNS , _______	Use the next lowest non-transparent key
KC_A		a and A
KC_B		b and B
KC_C		c and C
KC_D		d and D

```
{
{0,0,0,0},
{0,0,0,0},
{0,0,0,0},
{0,0,0,0},
{0,0,0,0}
}
#define LAYOUT( \
    k00, k01, k02, k03, \
    k10, k11, k12, k13, \
    k20, k21, k22, \
    k30, k31, k32, k33, \
    k40,      k42 \
) { \
    { k00, k01, k02, k03, }, \
    { k10, k11, k12, k13, }, \
    { k20, k21, k22, KC_NO, }, \
    { k30, k31, k32, k33, }, \
    { k40, KC_NO, k42, KC_NO } \
}
```

DIFFERENT METHODS FOR MODIFYING KEYBOARD SHORCUTS:

Windows:

- Download software to handle this for you
- Modifying the registry
- Drivers:
  - %RootDir%\system32
- Device Info:
  - %RootDir%\inf\*.inf
  - Vista: %RootDir%\system32\driverstore

LINUX:

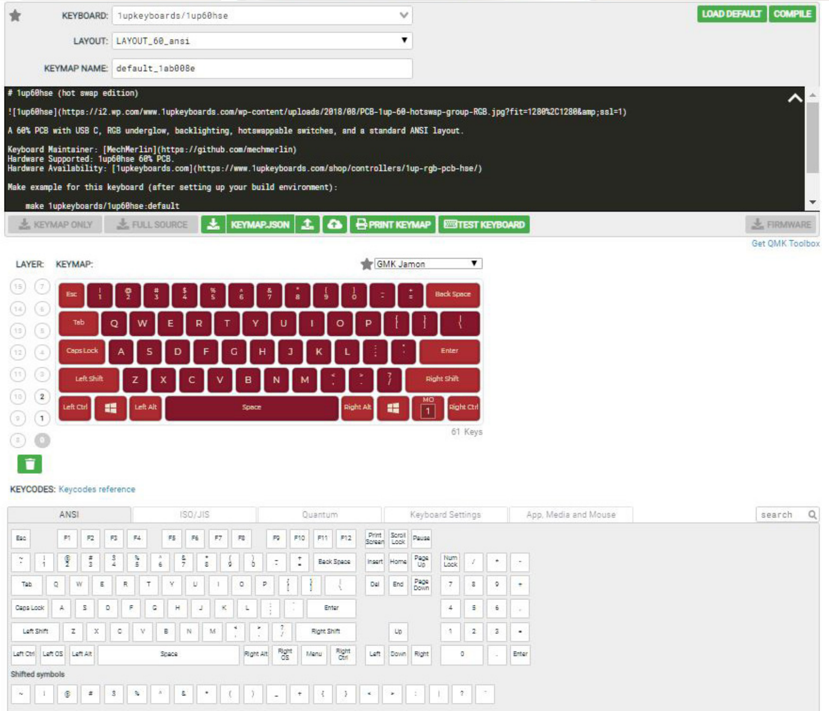
```
lsusb -v 2>&1 1>/dev/null
fdisk -l
lshw
/usr/share/misc/usb_hid_usages
/dev/input/eventX
/etc/udev/hwdb.d/*
showkey -s
showkey -scancodes
setkeycodes XXXX XXXX
xmodmap -pke
```

Updating and testing keymappings via kernel libraries:

```
udevadm hwdb --update
udevadm trigger --sysname-match=”event*”
evtest
```

QMK firmware and Tools:

Command Line, QMK Configurator/Toolbox, VIA



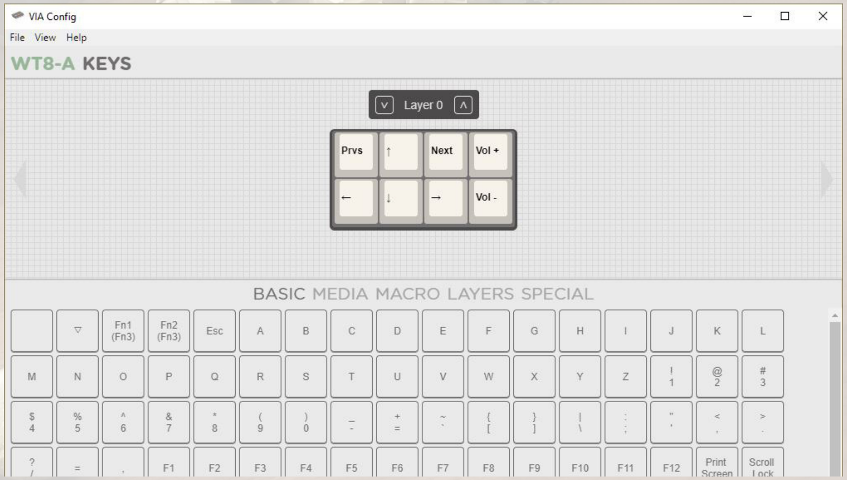
KEYBOARD: 1upkeyboards/1up60hse

LAYOUT: LAYOUT_60_ansi

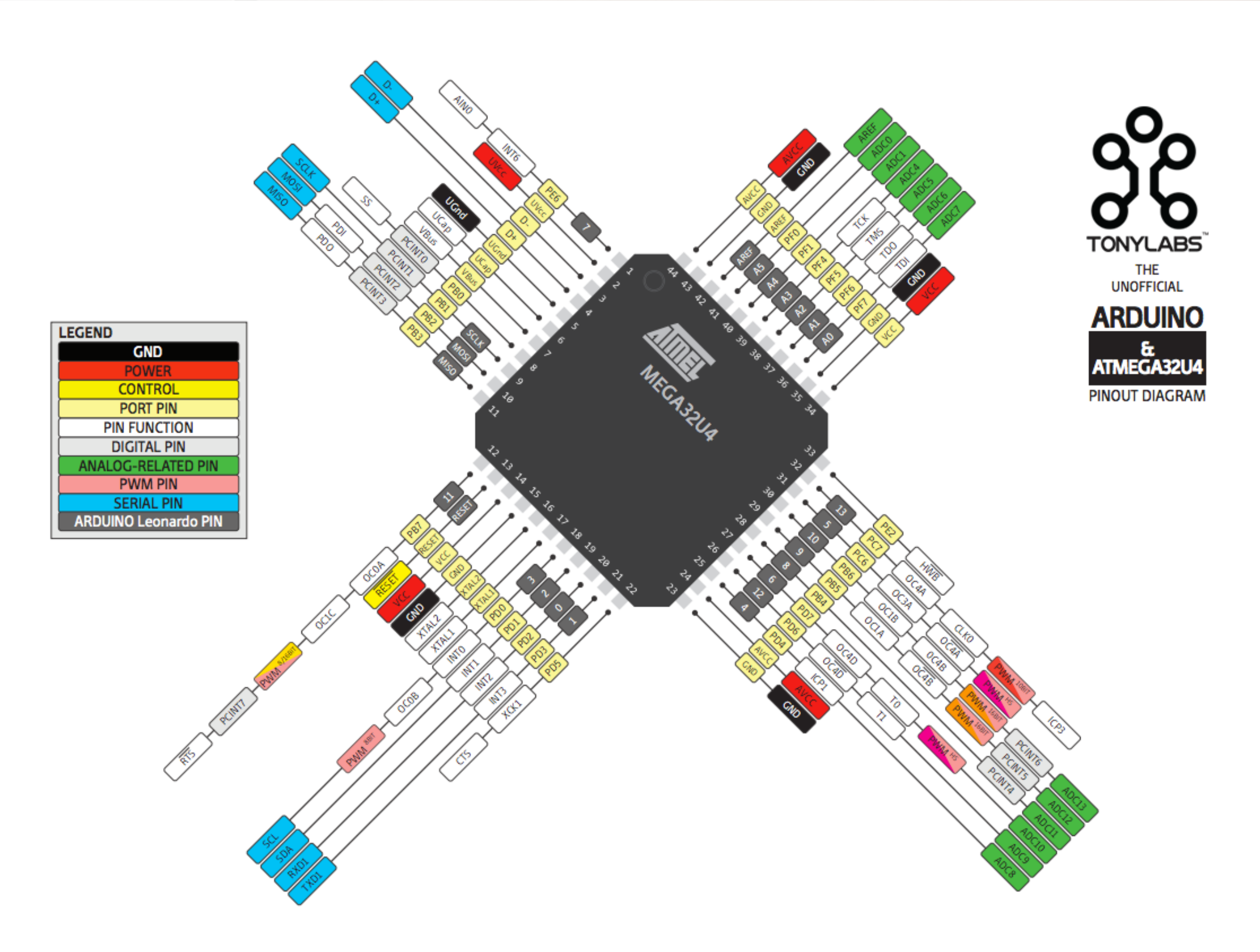
KEYMAP NAME: default_1ab008e

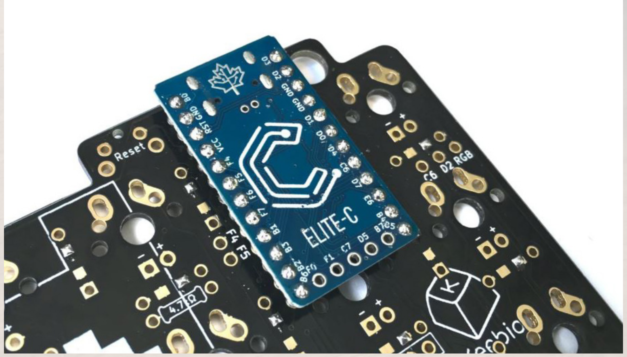
```
# 1up60hse (hot swap edition)

![1up60hse](https://i2.wp.com/www.1upkeyboards.com/wp-content/uploads/2018/08/PCB-1up-60-hotswap-group-RGB.jpg?fit=1280%2C1280&amp;ssl=1)

A 60% PCB with USB C, RGB underglow, backlighting, hotswappable switches, and a standard ANSI layout.

Keyboard Maintainer: [MechMerlin](https://github.com/mechmerlin)
Hardware Supported: 1up60hse 60% PCB.
Hardware Availability: [1upkeyboards.com](https://www.1upkeyboards.com/shop/controllers/1up-rgb-pcb-hse/)

Make example for this keyboard (after setting up your build environment):

    make 1upkeyboards/1up60hse:default
```

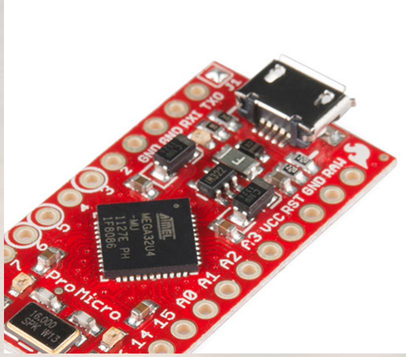
KEYMAP: GMK Jamon

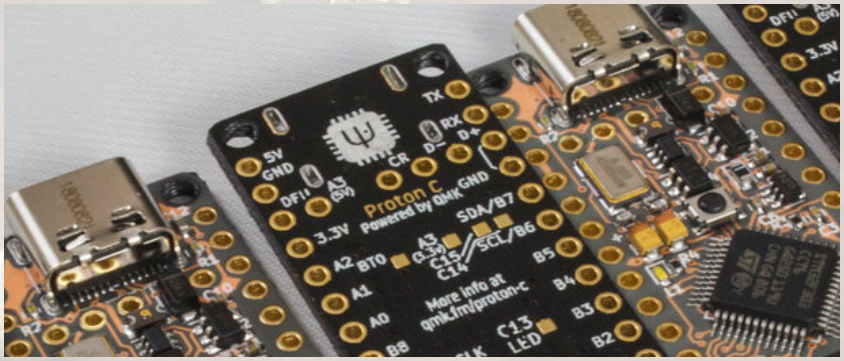


EXAMPLE MACROPAD THAT USES A MICROCONTROLLER RUNNING QMK:









FLASHING QMK & TOOLS:

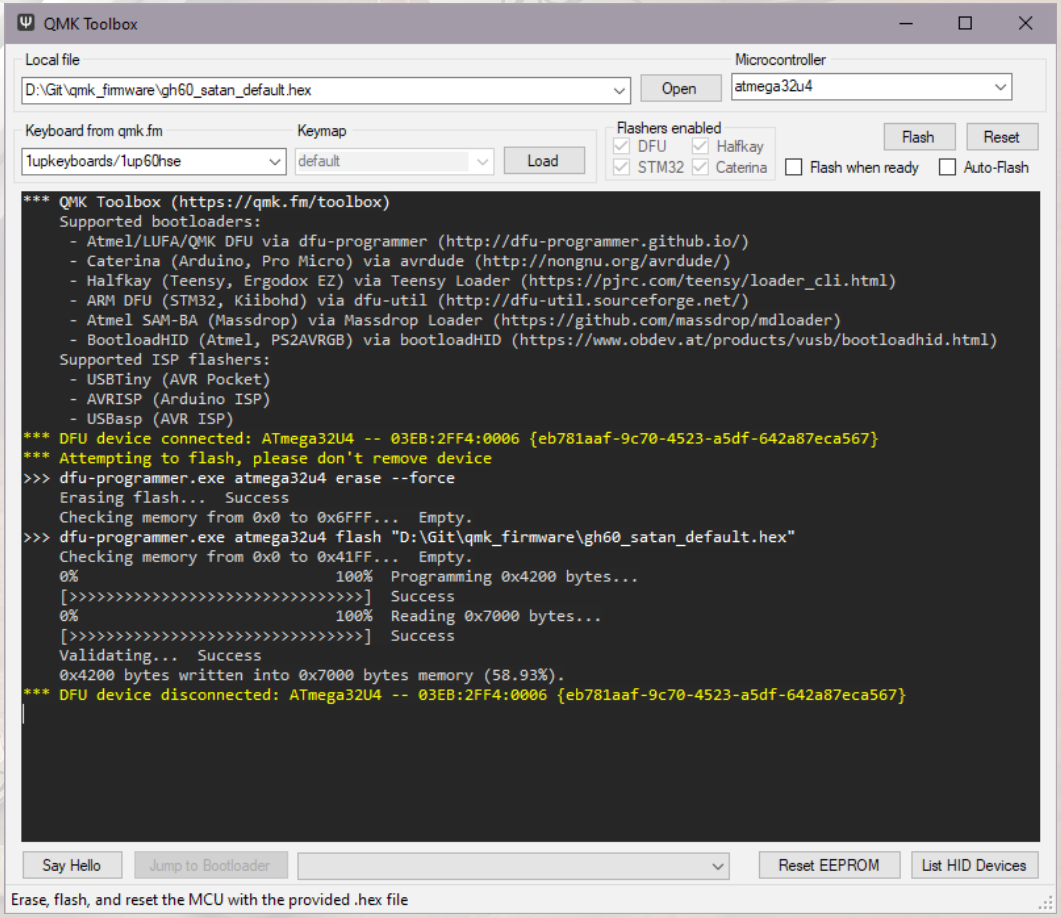
- Atmel's DFU bootloader comes on most atmega32u4 chips by default (dfu-programmer)
- "Created because the Atmel "FLIP" program for flashing devices does not support flashing via USB on Linux, and because standard DFU loaders do not work for Atmel's chips"
- QMK via Command Line, QMK Configurator / Toolbox, VIA Congifurator

STEPS TO FLASHING QMK:

1) Press the RESET keycode (mapped to keys)

   OR tap/short the RESET button (on board)

2) Wait for the OS to detect the board
3) Erase the memory
4) Flash a .hex file
5) Reset the device into application mode
6) make <keyboard>:<keymap>:dfu VERBOSE=true



Local file: D:\Git\qmk_firmware\gh60_satan_default.hex

Microcontroller: atmega32u4

Keyboard from qmk.fm: 1upkeyboards/1up60hse

Keymap: default

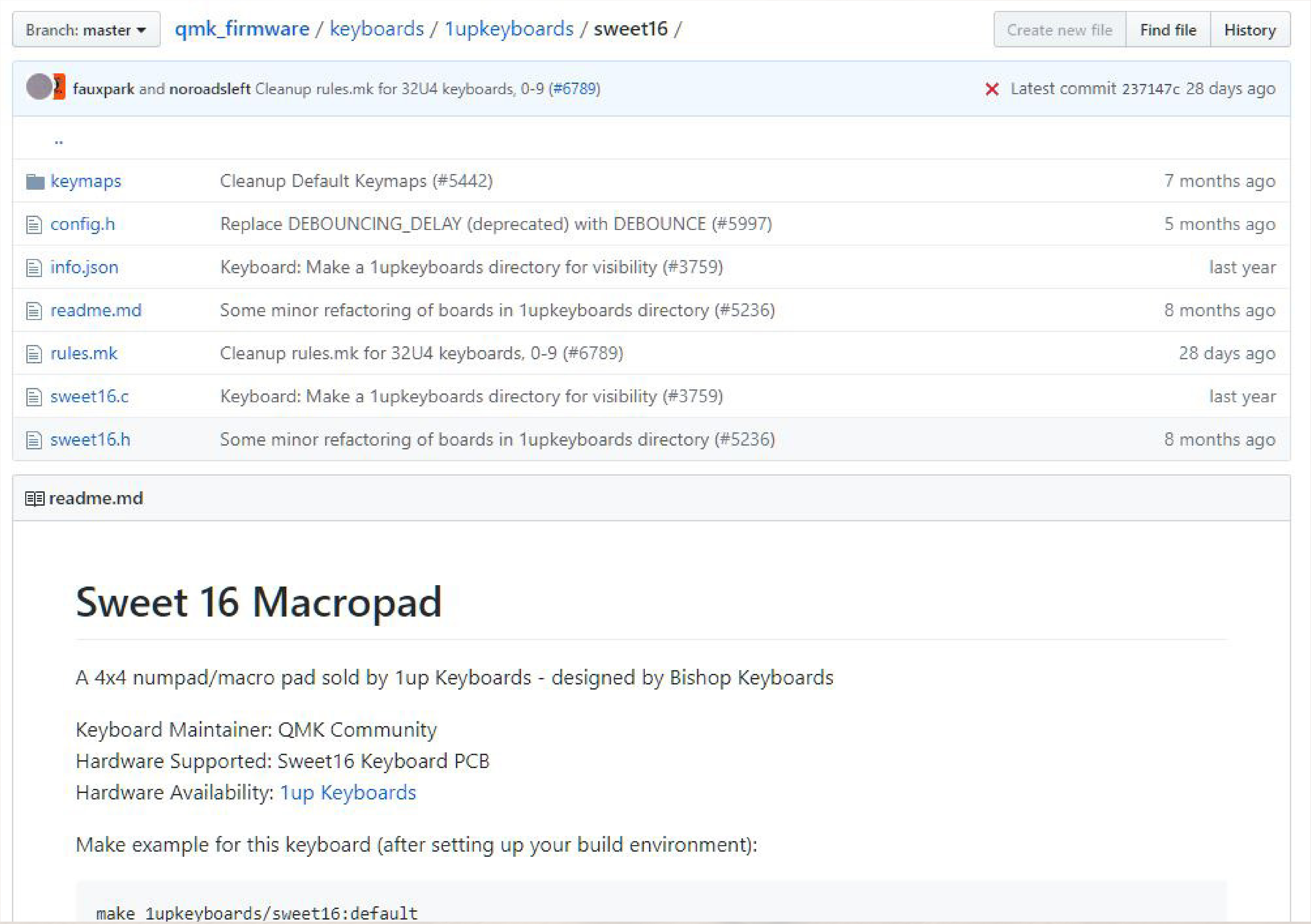
Flashers enabled: ☑ DFU ☑ Halfkay ☑ STM32 ☑ Caterina

□ Flash when ready □ Auto-Flash

```
*** QMK Toolbox (https://qmk.fm/toolbox)
    Supported bootloaders:
     - Atmel/LUFA/QMK DFU via dfu-programmer (http://dfu-programmer.github.io/)
     - Caterina (Arduino, Pro Micro) via avrdude (http://nongnu.org/avrdude/)
     - Halfkay (Teensy, Ergodox EZ) via Teensy Loader (https://pjrc.com/teensy/loader_cli.html)
     - ARM DFU (STM32, Kiibohd) via dfu-util (http://dfu-util.sourceforge.net/)
     - Atmel SAM-BA (Massdrop) via Massdrop Loader (https://github.com/massdrop/mdloader)
     - BootloadHID (Atmel, PS2AVRGB) via bootloadHID (https://www.obdev.at/products/vusb/bootloadhid.html)
    Supported ISP flashers:
     - USBTiny (AVR Pocket)
     - AVRISP (Arduino ISP)
     - USBasp (AVR ISP)
*** DFU device connected: ATmega32U4 -- 03EB:2FF4:0006 {eb781aaf-9c70-4523-a5df-642a87eca567}
*** Attempting to flash, please don't remove device
>>> dfu-programmer.exe atmega32u4 erase --force
    Erasing flash...  Success
    Checking memory from 0x0 to 0x6FFF...  Empty.
>>> dfu-programmer.exe atmega32u4 flash "D:\Git\qmk_firmware\gh60_satan_default.hex"
    Checking memory from 0x0 to 0x41FF...  Empty.
    0%                            100%  Programming 0x4200 bytes...
    [>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]  Success
    0%                            100%  Reading 0x7000 bytes...
    [>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]  Success
    Validating...  Success
    0x4200 bytes written into 0x7000 bytes memory (58.93%).
*** DFU device disconnected: ATmega32U4 -- 03EB:2FF4:0006 {eb781aaf-9c70-4523-a5df-642a87eca567}
```

Erase, flash, and reset the MCU with the provided .hex file

EXAMPLE QMK GITHUB ENTRY:



qmk_firmware / keyboards / 1upkeyboards / sweet16 /

fauxpark and noroadsleft Cleanup rules.mk for 32U4 keyboards, 0-9 (#6789) ✕ Latest commit 237147c 28 days ago

..		
keymaps	Cleanup Default Keymaps (#5442)	7 months ago
config.h	Replace DEBOUNCING_DELAY (deprecated) with DEBOUNCE (#5997)	5 months ago
info.json	Keyboard: Make a 1upkeyboards directory for visibility (#3759)	last year
readme.md	Some minor refactoring of boards in 1upkeyboards directory (#5236)	8 months ago
rules.mk	Cleanup rules.mk for 32U4 keyboards, 0-9 (#6789)	28 days ago
sweet16.c	Keyboard: Make a 1upkeyboards directory for visibility (#3759)	last year
sweet16.h	Some minor refactoring of boards in 1upkeyboards directory (#5236)	8 months ago

readme.md

Sweet 16 Macropad

A 4x4 numpad/macro pad sold by 1up Keyboards - designed by Bishop Keyboards

Keyboard Maintainer: QMK Community
Hardware Supported: Sweet16 Keyboard PCB
Hardware Availability: 1up Keyboards

Make example for this keyboard (after setting up your build environment):

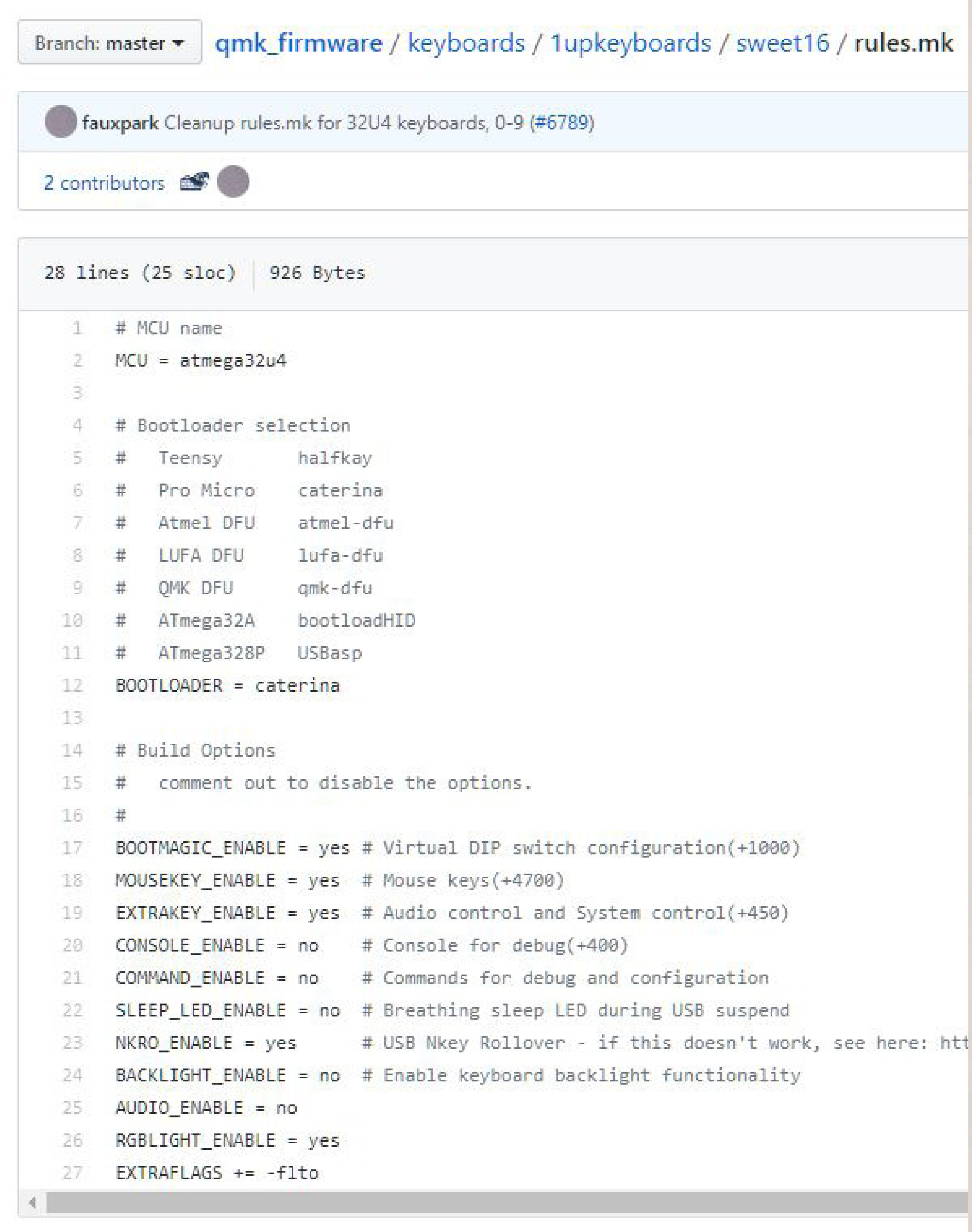
```
make 1upkeyboards/sweet16:default
```

QMK FILES EXPLAINED:

- **rules.mk**

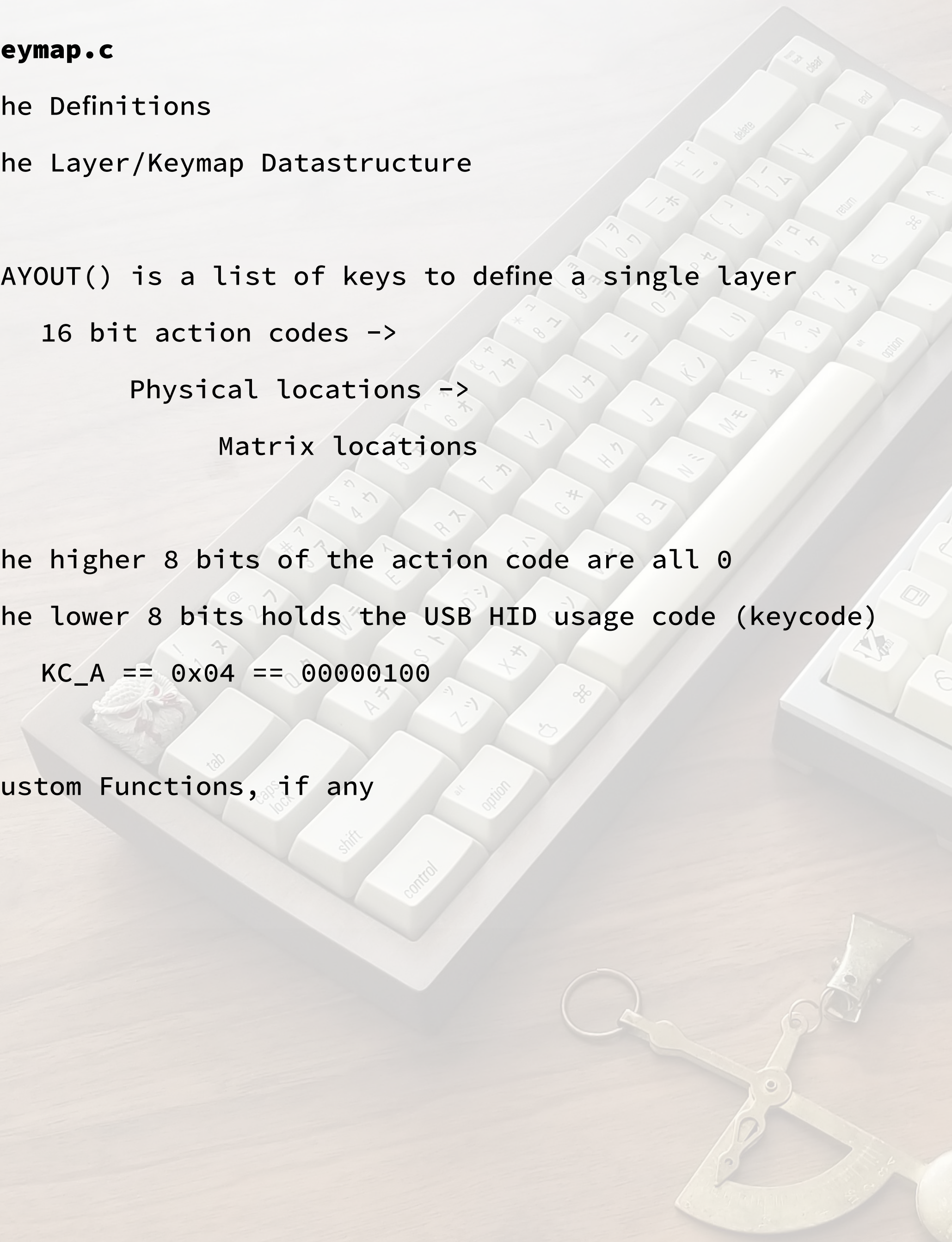
- Make file that is used to set information about the MCU (Microcontroller Unit that this firmware will be compiled to run on, as well as enabling or diabling some other build features)

- ‘Make’ is a utility that automatically determines which pieces of a large program need to be recompiled, and issues commands to recompile them


Branch: master ▾
qmk_firmware / keyboards / 1upkeyboards / sweet16 / rules.mk
fauxpark Cleanup rules.mk for 32U4 keyboards, 0-9 (#6789)
2 contributors
28 lines (25 sloc) | 926 Bytes

```
# MCU name
MCU = atmega32u4

# Bootloader selection
#   Teensy       halfkay
#   Pro Micro    caterina
#   Atmel DFU    atmel-dfu
#   LUFA DFU     lufa-dfu
#   QMK DFU      qmk-dfu
#   ATmega32A    bootloadHID
#   ATmega328P   USBasp
BOOTLOADER = caterina

# Build Options
#   comment out to disable the options.
#
BOOTMAGIC_ENABLE = yes # Virtual DIP switch configuration(+1000)
MOUSEKEY_ENABLE = yes  # Mouse keys(+4700)
EXTRAKEY_ENABLE = yes  # Audio control and System control(+450)
CONSOLE_ENABLE = no    # Console for debug(+400)
COMMAND_ENABLE = no    # Commands for debug and configuration
SLEEP_LED_ENABLE = no  # Breathing sleep LED during USB suspend
NKRO_ENABLE = yes      # USB Nkey Rollover - if this doesn't work, see here: htt
BACKLIGHT_ENABLE = no  # Enable keyboard backlight functionality
AUDIO_ENABLE = no
RGBLIGHT_ENABLE = yes
EXTRAFLAGS += -flto
```

QMK FILES EXPLAINED:



- **keymap.c**
- The Definitions
- The Layer/Keymap Datastructure

- LAYOUT() is a list of keys to define a single layer

    16 bit action codes ->

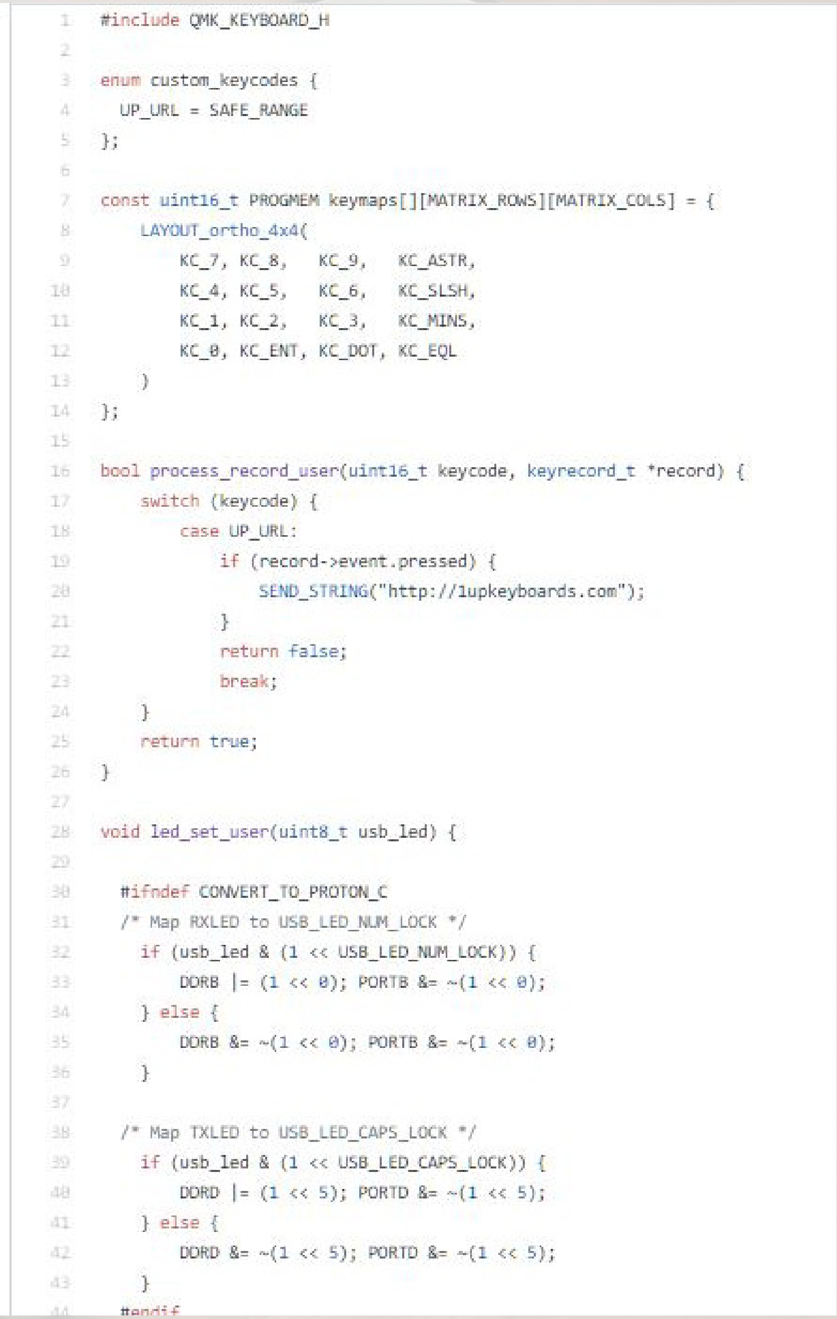
        Physical locations ->

            Matrix locations

- The higher 8 bits of the action code are all 0
- The lower 8 bits holds the USB HID usage code (keycode)

    KC_A == 0x04 == 00000100

- Custom Functions, if any



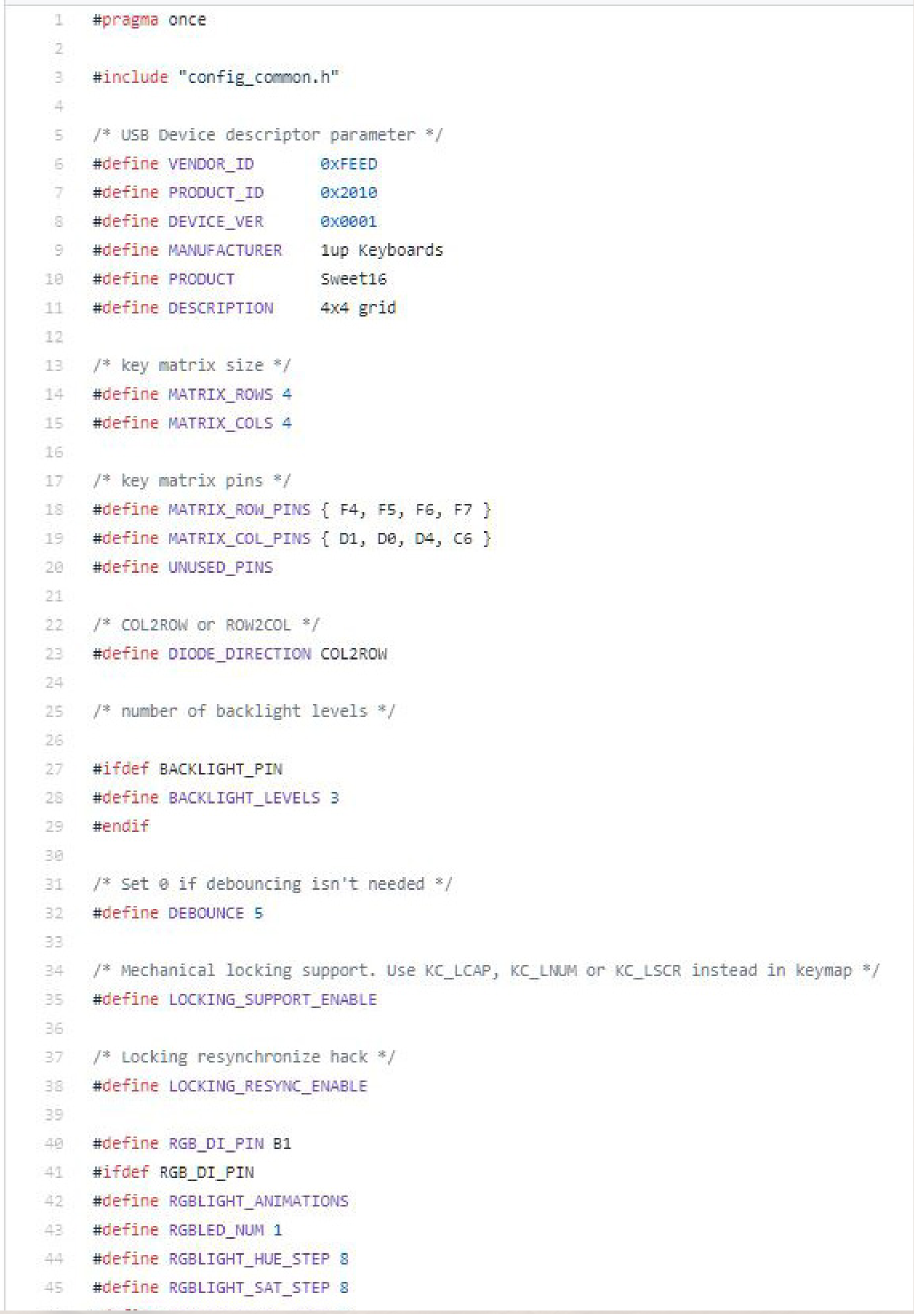
```
#include QMK_KEYBOARD_H

enum custom_keycodes {
  UP_URL = SAFE_RANGE
};

const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
    LAYOUT_ortho_4x4(
        KC_7, KC_8,   KC_9,   KC_ASTR,
        KC_4, KC_5,   KC_6,   KC_SLSH,
        KC_1, KC_2,   KC_3,   KC_MINS,
        KC_0, KC_ENT, KC_DOT, KC_EQL
    )
};

bool process_record_user(uint16_t keycode, keyrecord_t *record) {
    switch (keycode) {
        case UP_URL:
            if (record->event.pressed) {
                SEND_STRING("http://1upkeyboards.com");
            }
            return false;
            break;
    }
    return true;
}

void led_set_user(uint8_t usb_led) {

  #ifndef CONVERT_TO_PROTON_C
  /* Map RXLED to USB_LED_NUM_LOCK */
    if (usb_led & (1 << USB_LED_NUM_LOCK)) {
        DDRB |= (1 << 0); PORTB &= ~(1 << 0);
    } else {
        DDRB &= ~(1 << 0); PORTB &= ~(1 << 0);
    }

  /* Map TXLED to USB_LED_CAPS_LOCK */
    if (usb_led & (1 << USB_LED_CAPS_LOCK)) {
        DDRD |= (1 << 5); PORTD &= ~(1 << 5);
    } else {
        DDRD &= ~(1 << 5); PORTD &= ~(1 << 5);
    }
  #endif
```

QMK FILES EXPLAINED:

- **config.h**

- C header file that contains device descriptors such as hardware options, features and behaviors.

- This file sets things like the matrix size, product name, USB VID/PID, description and other settings, as well as other defaults to ensure your board is always working.

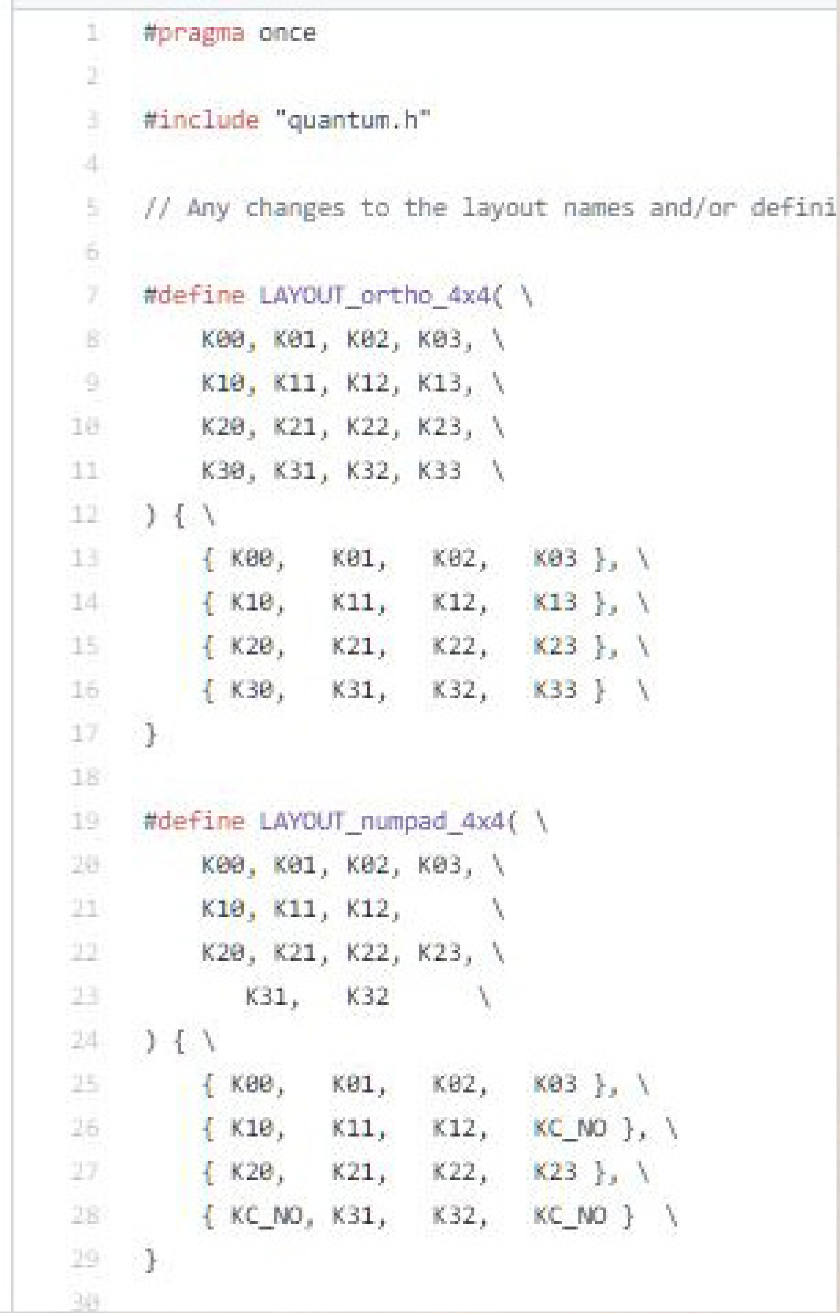


```
#pragma once

#include "config_common.h"

/* USB Device descriptor parameter */
#define VENDOR_ID       0xFEED
#define PRODUCT_ID      0x2010
#define DEVICE_VER      0x0001
#define MANUFACTURER    1up Keyboards
#define PRODUCT         Sweet16
#define DESCRIPTION     4x4 grid

/* key matrix size */
#define MATRIX_ROWS 4
#define MATRIX_COLS 4

/* key matrix pins */
#define MATRIX_ROW_PINS { F4, F5, F6, F7 }
#define MATRIX_COL_PINS { D1, D0, D4, C6 }
#define UNUSED_PINS

/* COL2ROW or ROW2COL */
#define DIODE_DIRECTION COL2ROW

/* number of backlight levels */

#ifdef BACKLIGHT_PIN
#define BACKLIGHT_LEVELS 3
#endif

/* Set 0 if debouncing isn't needed */
#define DEBOUNCE 5

/* Mechanical locking support. Use KC_LCAP, KC_LNUM or KC_LSCR instead in keymap */
#define LOCKING_SUPPORT_ENABLE

/* Locking resynchronize hack */
#define LOCKING_RESYNC_ENABLE

#define RGB_DI_PIN B1
#ifdef RGB_DI_PIN
#define RGBLIGHT_ANIMATIONS
#define RGBLED_NUM 1
#define RGBLIGHT_HUE_STEP 8
#define RGBLIGHT_SAT_STEP 8
```

QMK FILES EXPLAINED:

- keymap.h

- File that contains the physical mapping of pins and the matrix definitions for that mapping
- Two parts
- First half is an ordered array of matrix positions by switch positions (this maps directly to keymap)
- Second half is the two-dimensional array that defines the matrix and pairs it to a matrix position code.



```
#pragma once

#include "quantum.h"

// Any changes to the layout names and/or defini

#define LAYOUT_ortho_4x4( \
    K00, K01, K02, K03, \
    K10, K11, K12, K13, \
    K20, K21, K22, K23, \
    K30, K31, K32, K33  \
) { \
    { K00,   K01,   K02,   K03 }, \
    { K10,   K11,   K12,   K13 }, \
    { K20,   K21,   K22,   K23 }, \
    { K30,   K31,   K32,   K33 }  \
}

#define LAYOUT_numpad_4x4( \
    K00, K01, K02, K03, \
    K10, K11, K12,      \
    K20, K21, K22, K23, \
       K31,   K32      \
) { \
    { K00,   K01,   K02,   K03 }, \
    { K10,   K11,   K12,   KC_NO }, \
    { K20,   K21,   K22,   K23 }, \
    { KC_NO, K31,   K32,   KC_NO }  \
}
```

- Matrix Scanning abriged

1) Matrix scanning loop runs checking for changes (0 or 1)

2) Keypress is detected at a defined location (K00)

3) Loop locates [0,0] in matrix and finds what key identifer it corresponds to in the layout macro

4) In layout macro K00 will equal a keycode

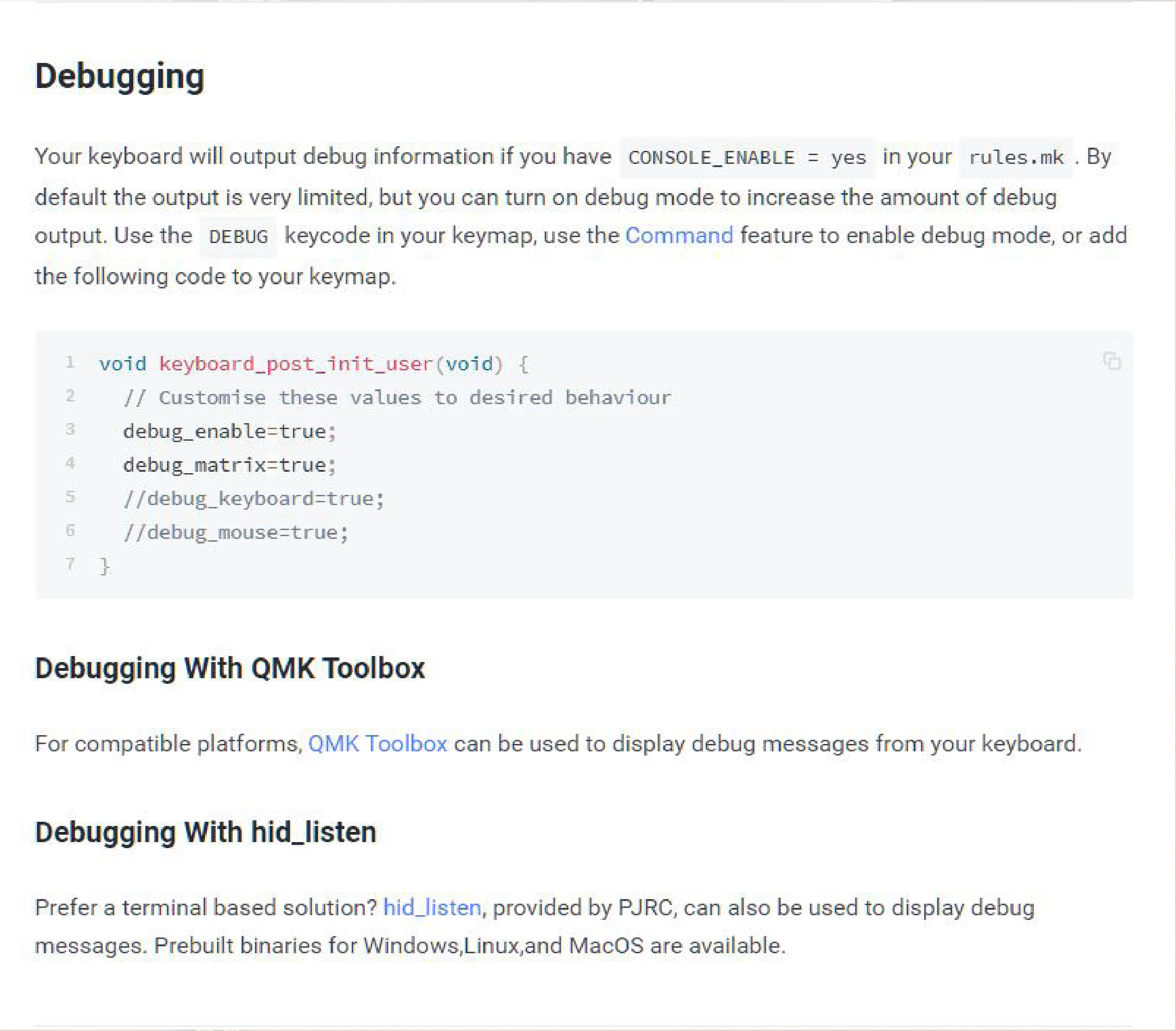
5) Keycode is sent to operating system, or an action is taken such as switching layers, changing lighting, or changing modes such as bootloader or entering debubg (more on this later).

LET’S TALK ABOUT DEBUGGING AND FEATURES:

- Finding and removing errors from hardware or software.

- Normally a feature you enable
- Frequently left on by accident
- Often grants elevated access or privileges into the hardware or software that is being debugged

- In our case, QMK allows for everything from keycodes to custom printed error messages, all user defined



Debugging

Your keyboard will output debug information if you have `CONSOLE_ENABLE = yes` in your `rules.mk`. By default the output is very limited, but you can turn on debug mode to increase the amount of debug output. Use the `DEBUG` keycode in your keymap, use the Command feature to enable debug mode, or add the following code to your keymap.

```
void keyboard_post_init_user(void) {
  // Customise these values to desired behaviour
  debug_enable=true;
  debug_matrix=true;
  //debug_keyboard=true;
  //debug_mouse=true;
}
```

Debugging With QMK Toolbox

For compatible platforms, QMK Toolbox can be used to display debug messages from your keyboard.

Debugging With hid_listen

Prefer a terminal based solution? hid_listen, provided by PJRC, can also be used to display debug messages. Prebuilt binaries for Windows,Linux,and MacOS are available.

EDITING FILES TO USE DEBUG FEATURES IN QMK:

FILE EDITS IN BLUE:

Left: keymap.c

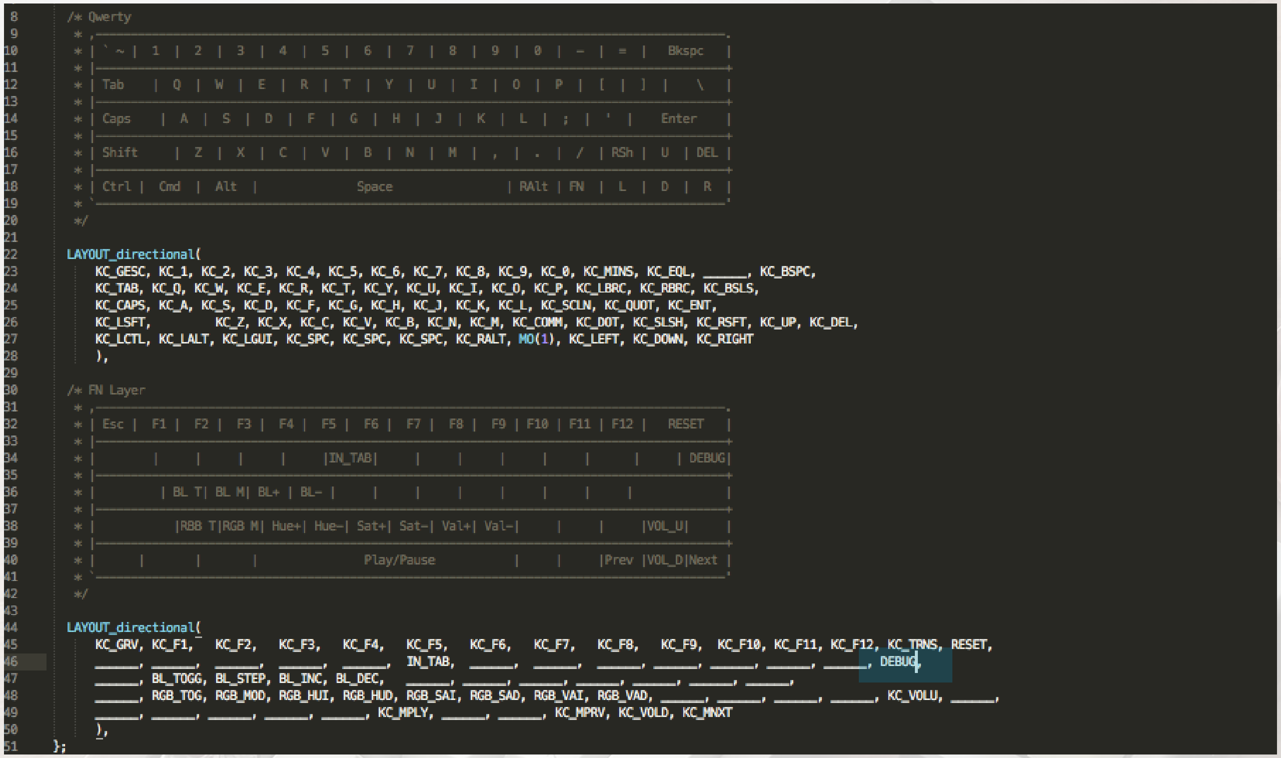
(add debug keycode)

Bottom Left: rules.mk

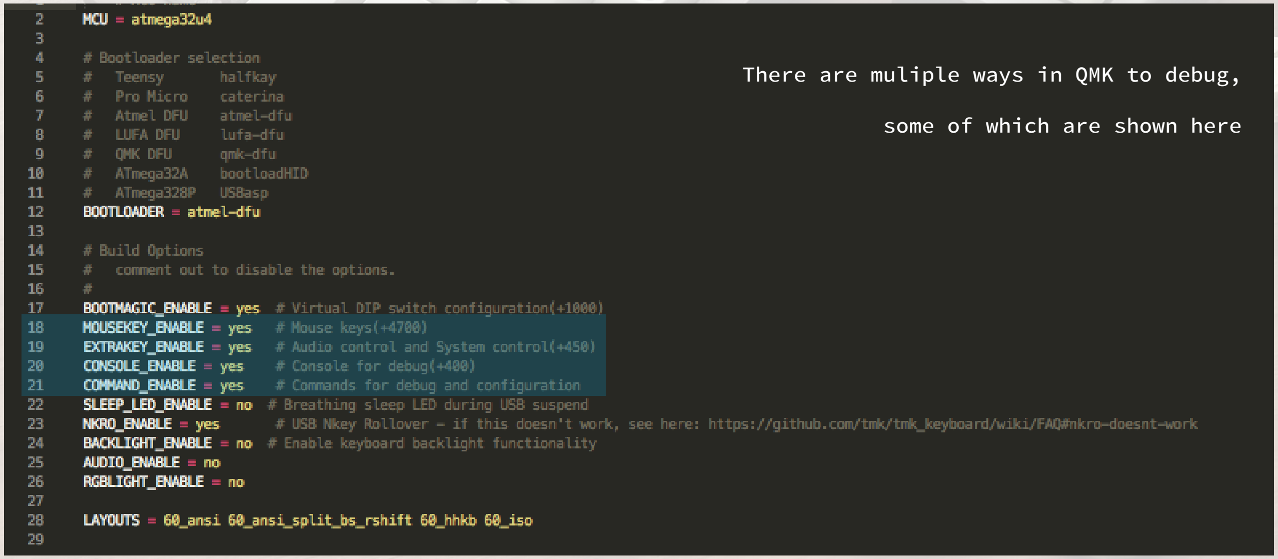
(enable debug capability)

Right: dz60.c

(print out debug messages)



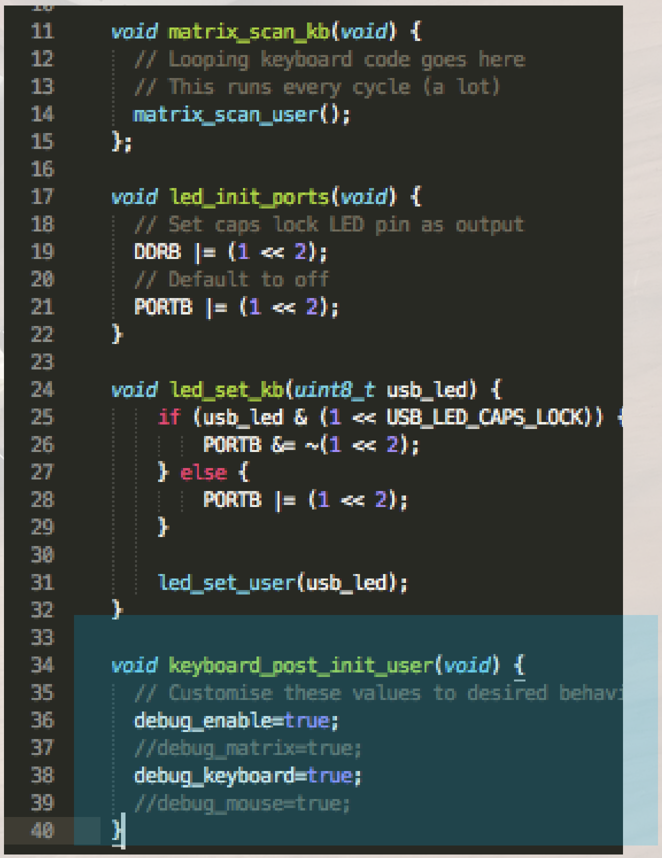
```
/* Qwerty
 * ,-----------------------------------------------------------------------------------------.
 * | ` ~ |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  0  |  -  |  =  |   Bkspc   |
 * |-----------------------------------------------------------------------------------------+
 * | Tab    |  Q  |  W  |  E  |  R  |  T  |  Y  |  U  |  I  |  O  |  P  |  [  |  ]  |    \   |
 * |-----------------------------------------------------------------------------------------+
 * | Caps    |  A  |  S  |  D  |  F  |  G  |  H  |  J  |  K  |  L  |  ;  |  '  |    Enter   |
 * |-----------------------------------------------------------------------------------------+
 * | Shift     |  Z  |  X  |  C  |  V  |  B  |  N  |  M  |  ,  |  .  |  /  | RSh |  U  | DEL |
 * |-----------------------------------------------------------------------------------------+
 * | Ctrl |  Cmd  |  Alt  |               Space                  | RAlt | FN  |  L  |  D  |  R  |
 * `-----------------------------------------------------------------------------------------'
 */

LAYOUT_directional(
    KC_GESC, KC_1, KC_2, KC_3, KC_4, KC_5, KC_6, KC_7, KC_8, KC_9, KC_0, KC_MINS, KC_EQL, _______, KC_BSPC,
    KC_TAB, KC_Q, KC_W, KC_E, KC_R, KC_T, KC_Y, KC_U, KC_I, KC_O, KC_P, KC_LBRC, KC_RBRC, KC_BSLS,
    KC_CAPS, KC_A, KC_S, KC_D, KC_F, KC_G, KC_H, KC_J, KC_K, KC_L, KC_SCLN, KC_QUOT, KC_ENT,
    KC_LSFT,         KC_Z, KC_X, KC_C, KC_V, KC_B, KC_N, KC_M, KC_COMM, KC_DOT, KC_SLSH, KC_RSFT, KC_UP, KC_DEL,
    KC_LCTL, KC_LALT, KC_LGUI, KC_SPC, KC_SPC, KC_SPC, KC_RALT, MO(1), KC_LEFT, KC_DOWN, KC_RIGHT
    ),

/* FN Layer
 * ,-----------------------------------------------------------------------------------------.
 * | Esc |  F1 |  F2 |  F3 |  F4 |  F5 |  F6 |  F7 |  F8 |  F9 | F10 | F11 | F12 |   RESET   |
 * |-----------------------------------------------------------------------------------------+
 * |        |     |     |     |     |IN_TAB|    |     |     |     |     |     |     | DEBUG|
 * |-----------------------------------------------------------------------------------------+
 * |         | BL T| BL M| BL+ | BL- |     |     |     |     |     |     |     |            |
 * |-----------------------------------------------------------------------------------------+
 * |           |RBB T|RGB M| Hue+| Hue-| Sat+| Sat-| Val+| Val-|     |     |     |VOL_U|     |
 * |-----------------------------------------------------------------------------------------+
 * |      |      |      |              Play/Pause              |      |      |Prev |VOL_D|Next |
 * `-----------------------------------------------------------------------------------------'
 */

LAYOUT_directional(
    KC_GRV, KC_F1,   KC_F2,   KC_F3,   KC_F4,   KC_F5,   KC_F6,   KC_F7,   KC_F8,   KC_F9,   KC_F10, KC_F11, KC_F12, KC_TRNS, RESET,
    _______, _______, _______, _______, _______, IN_TAB, _______, _______, _______, _______, _______, _______, _______, DEBUG,
    _______, BL_TOGG, BL_STEP, BL_INC, BL_DEC,  _______, _______, _______, _______, _______, _______, _______,
    _______, RGB_TOG, RGB_MOD, RGB_HUI, RGB_HUD, RGB_SAI, RGB_SAD, RGB_VAI, RGB_VAD, _______, _______, _______, _______, KC_VOLU, _______,
    _______, _______, _______, _______, _______, KC_MPLY, _______, _______, KC_MPRV, KC_VOLD, KC_MNXT
    ),
};
```



```
MCU = atmega32u4

# Bootloader selection
#   Teensy       halfkay
#   Pro Micro    caterina
#   Atmel DFU    atmel-dfu
#   LUFA DFU     lufa-dfu
#   QMK DFU      qmk-dfu
#   ATmega32A    bootloadHID
#   ATmega328P   USBasp
BOOTLOADER = atmel-dfu

# Build Options
#   comment out to disable the options.
#
BOOTMAGIC_ENABLE = yes  # Virtual DIP switch configuration(+1000)
MOUSEKEY_ENABLE = yes   # Mouse keys(+4700)
EXTRAKEY_ENABLE = yes   # Audio control and System control(+450)
CONSOLE_ENABLE = yes    # Console for debug(+400)
COMMAND_ENABLE = yes    # Commands for debug and configuration
SLEEP_LED_ENABLE = no  # Breathing sleep LED during USB suspend
NKRO_ENABLE = yes       # USB Nkey Rollover - if this doesn't work, see here: https://github.com/tmk/tmk_keyboard/wiki/FAQ#nkro-doesnt-work
BACKLIGHT_ENABLE = no  # Enable keyboard backlight functionality
AUDIO_ENABLE = no
RGBLIGHT_ENABLE = no

LAYOUTS = 60_ansi 60_ansi_split_bs_rshift 60_hhkb 60_iso

```

There are muliple ways in QMK to debug,

some of which are shown here

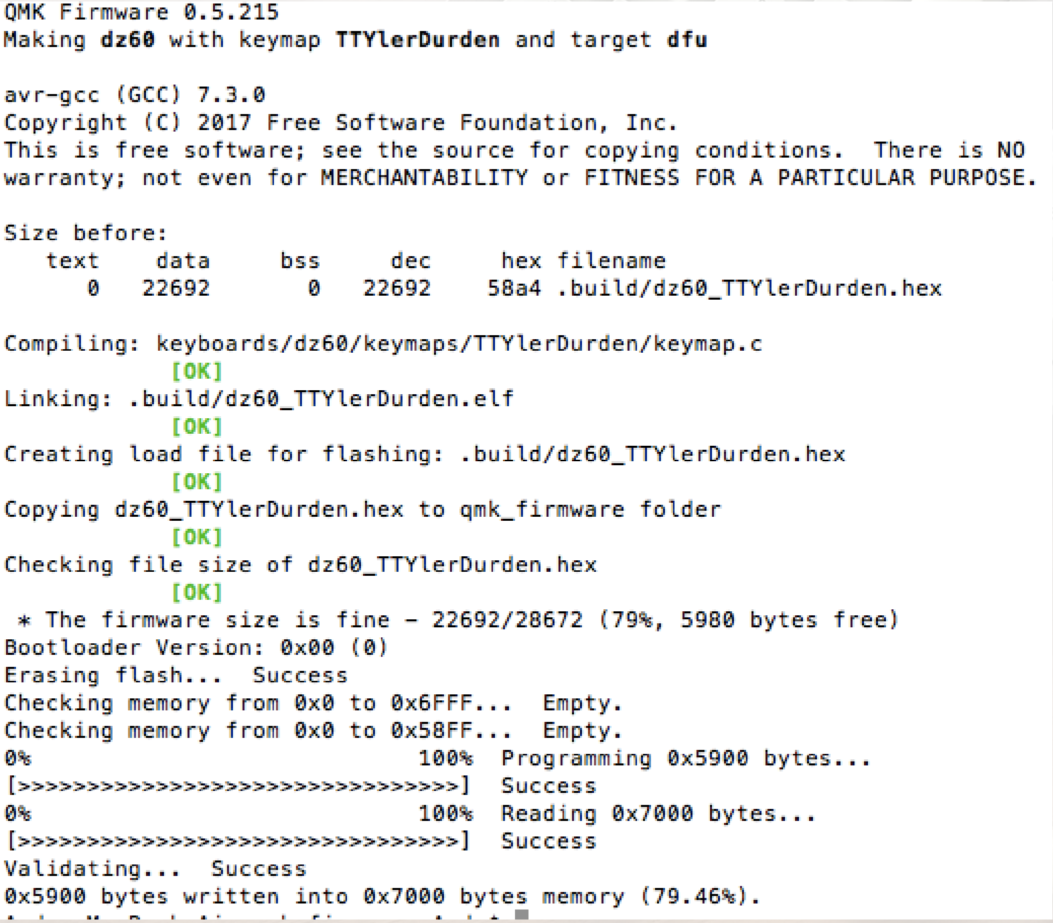


```c
void matrix_scan_kb(void) {
  // Looping keyboard code goes here
  // This runs every cycle (a lot)
  matrix_scan_user();
};

void led_init_ports(void) {
  // Set caps lock LED pin as output
  DDRB |= (1 << 2);
  // Default to off
  PORTB |= (1 << 2);
}

void led_set_kb(uint8_t usb_led) {
    if (usb_led & (1 << USB_LED_CAPS_LOCK)) {
        PORTB &= ~(1 << 2);
    } else {
        PORTB |= (1 << 2);
    }

    led_set_user(usb_led);
}

void keyboard_post_init_user(void) {
  // Customise these values to desired behavi
  debug_enable=true;
  //debug_matrix=true;
  debug_keyboard=true;
  //debug_mouse=true;
}
```

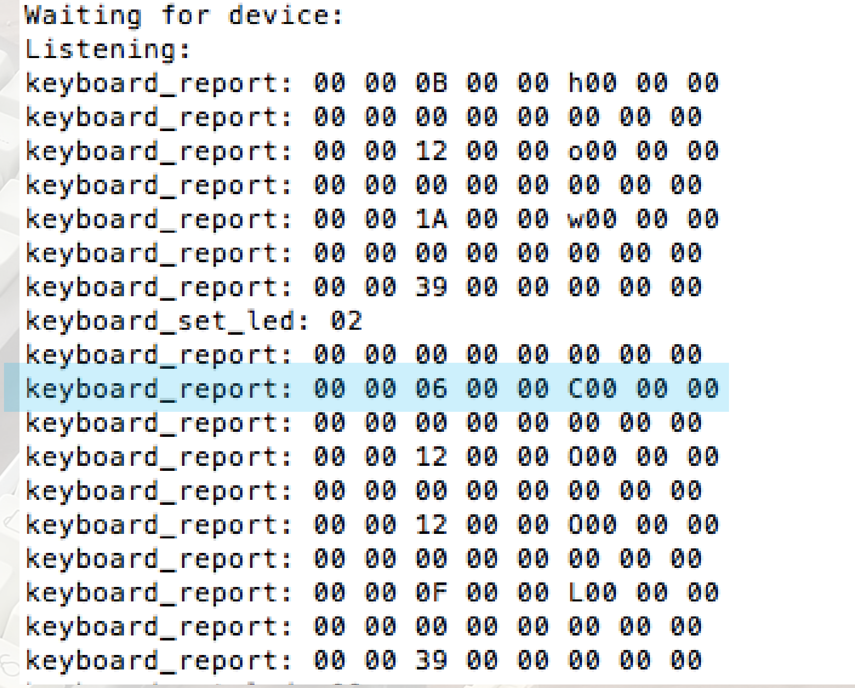
FLASHING KEYMAP AND TESTING DEBUG/KEYLOGGING:

- Make firmware file to flash onto board that now enable the use of debug messages to be printed
- Run hid_listen to print debug messages which are currently printing bitmask and keycode (in blue)

  KEY_H 0x0b // Keyboard h and H



```
QMK Firmware 0.5.215
Making dz60 with keymap TTYlerDurden and target dfu

avr-gcc (GCC) 7.3.0
Copyright (C) 2017 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   22692       0   22692    58a4 .build/dz60_TTYlerDurden.hex

Compiling: keyboards/dz60/keymaps/TTYlerDurden/keymap.c
            [OK]
Linking: .build/dz60_TTYlerDurden.elf
            [OK]
Creating load file for flashing: .build/dz60_TTYlerDurden.hex
            [OK]
Copying dz60_TTYlerDurden.hex to qmk_firmware folder
            [OK]
Checking file size of dz60_TTYlerDurden.hex
            [OK]
 * The firmware size is fine - 22692/28672 (79%, 5980 bytes free)
Bootloader Version: 0x00 (0)
Erasing flash...  Success
Checking memory from 0x0 to 0x6FFF...  Empty.
Checking memory from 0x0 to 0x58FF...  Empty.
0%                            100%  Programming 0x5900 bytes...
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]  Success
0%                            100%  Reading 0x7000 bytes...
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]  Success
Validating...  Success
0x5900 bytes written into 0x7000 bytes memory (79.46%).
```



```
Waiting for device:
Listening:
keyboard_report: 00 00 0B 00 00 h00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 12 00 00 o00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 1A 00 00 w00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 39 00 00 00 00 00
keyboard_set_led: 02
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 06 00 00 C00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 12 00 00 O00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 12 00 00 O00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 0F 00 00 L00 00 00
keyboard_report: 00 00 00 00 00 00 00 00
keyboard_report: 00 00 39 00 00 00 00 00
```

This output is slightly misleading, as the pressed key is being added into the keyboard_report output. For example:

keyboard_report: 00 00 06 00 00 C00 00 00

is really

keyboard_report: 02 00 06 00 00 00 00 00

since ‘C’ is actually ‘LSHIFT + C’

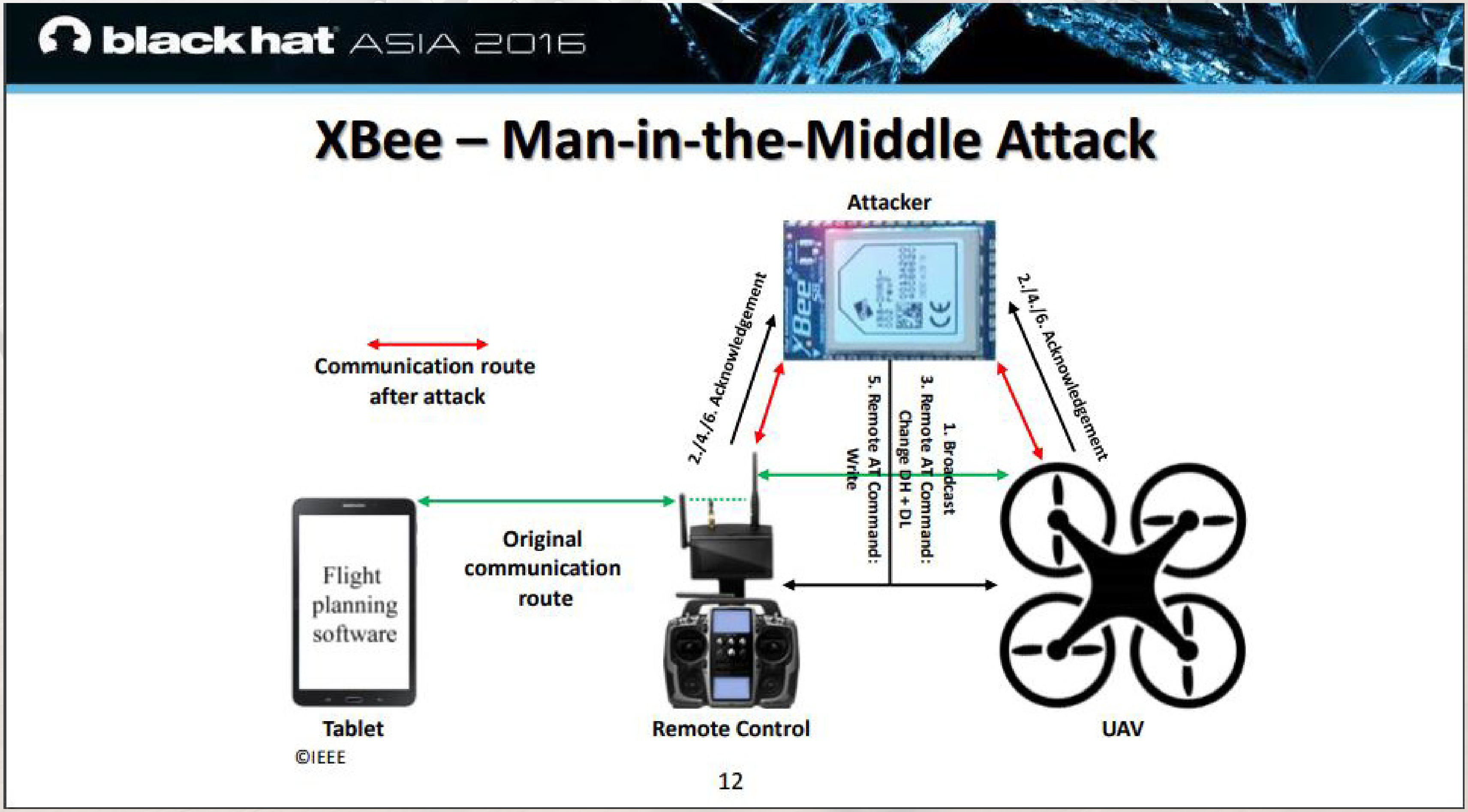
and **0x20** instead if it was ‘RSHIFT + C’

REAL WORLD SCENARIOS:

Blackhat 2016 SINGAPORE

Nils Rodday finds MiTM Capability in XBee chips (used in mesh networks) via specially crafted packets, reassociating the UAV to the attacker.

1) API Mode
2) Broadcast Feature
3) Remote AT Commands

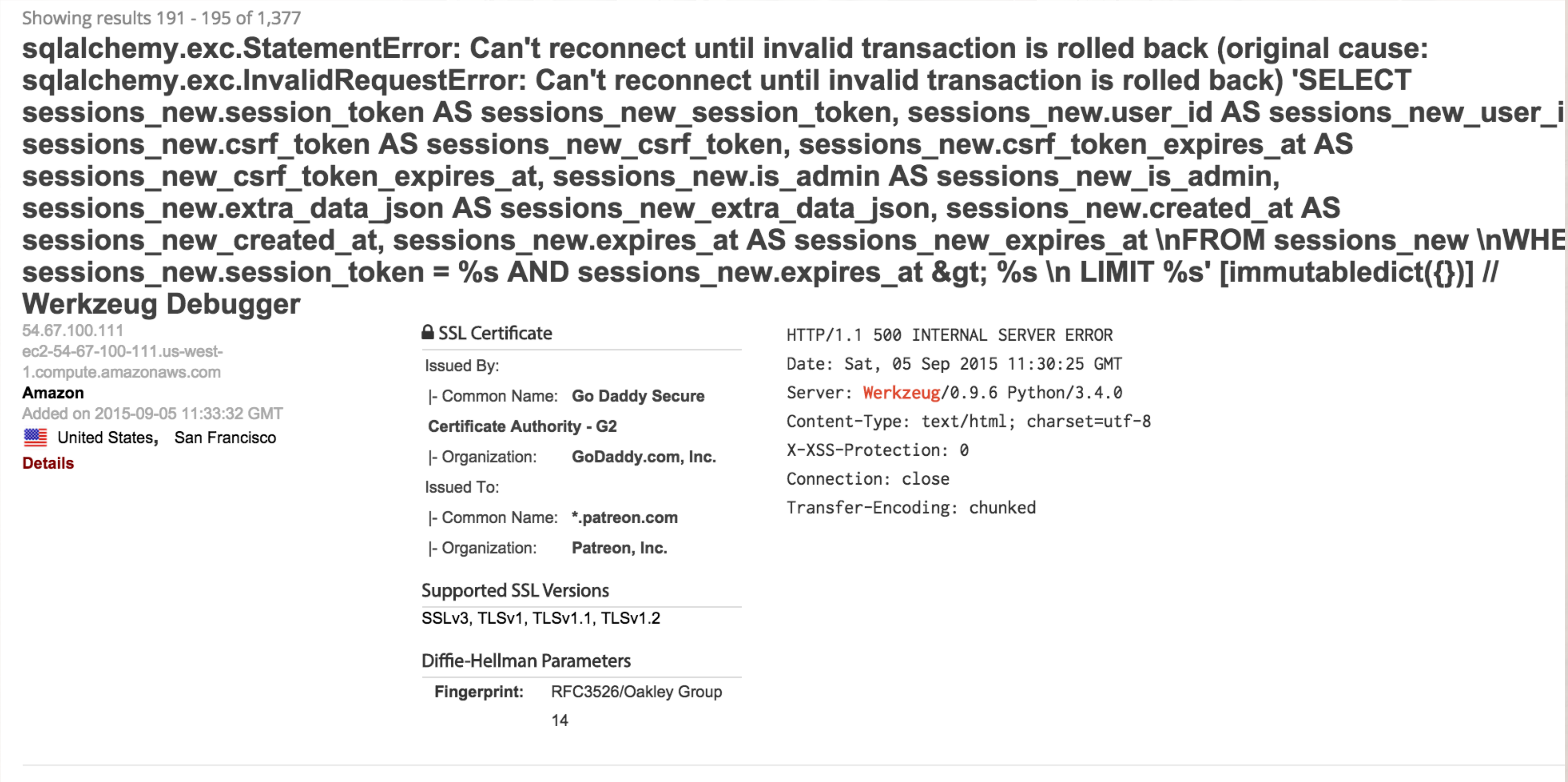


XBee – Man-in-the-Middle Attack

REAL WORLD SCENARIOS:

2015 Patreon is hacked via Werkzeuf Debugger RCE

Detectify labs alerts Patreon to an RCE they were vulnerable to after the debugger was left exposed to the internet (displayed below by shodan results).



Showing results 191 - 195 of 1,377

sqlalchemy.exc.StatementError: Can't reconnect until invalid transaction is rolled back (original cause: sqlalchemy.exc.InvalidRequestError: Can't reconnect until invalid transaction is rolled back) 'SELECT sessions_new.session_token AS sessions_new_session_token, sessions_new.user_id AS sessions_new_user_i sessions_new.csrf_token AS sessions_new_csrf_token, sessions_new.csrf_token_expires_at AS sessions_new_csrf_token_expires_at, sessions_new.is_admin AS sessions_new_is_admin, sessions_new.extra_data_json AS sessions_new_extra_data_json, sessions_new.created_at AS sessions_new_created_at, sessions_new.expires_at AS sessions_new_expires_at \nFROM sessions_new \nWHE sessions_new.session_token = %s AND sessions_new.expires_at &gt; %s \n LIMIT %s' [immutabledict({})] // Werkzeug Debugger

54.67.100.111
ec2-54-67-100-111.us-west-1.compute.amazonaws.com
Amazon
Added on 2015-09-05 11:33:32 GMT
United States, San Francisco
Details

🔒 SSL Certificate

Issued By:
|- Common Name: **Go Daddy Secure Certificate Authority - G2**
|- Organization: **GoDaddy.com, Inc.**
Issued To:
|- Common Name: ***.patreon.com**
|- Organization: **Patreon, Inc.**

Supported SSL Versions

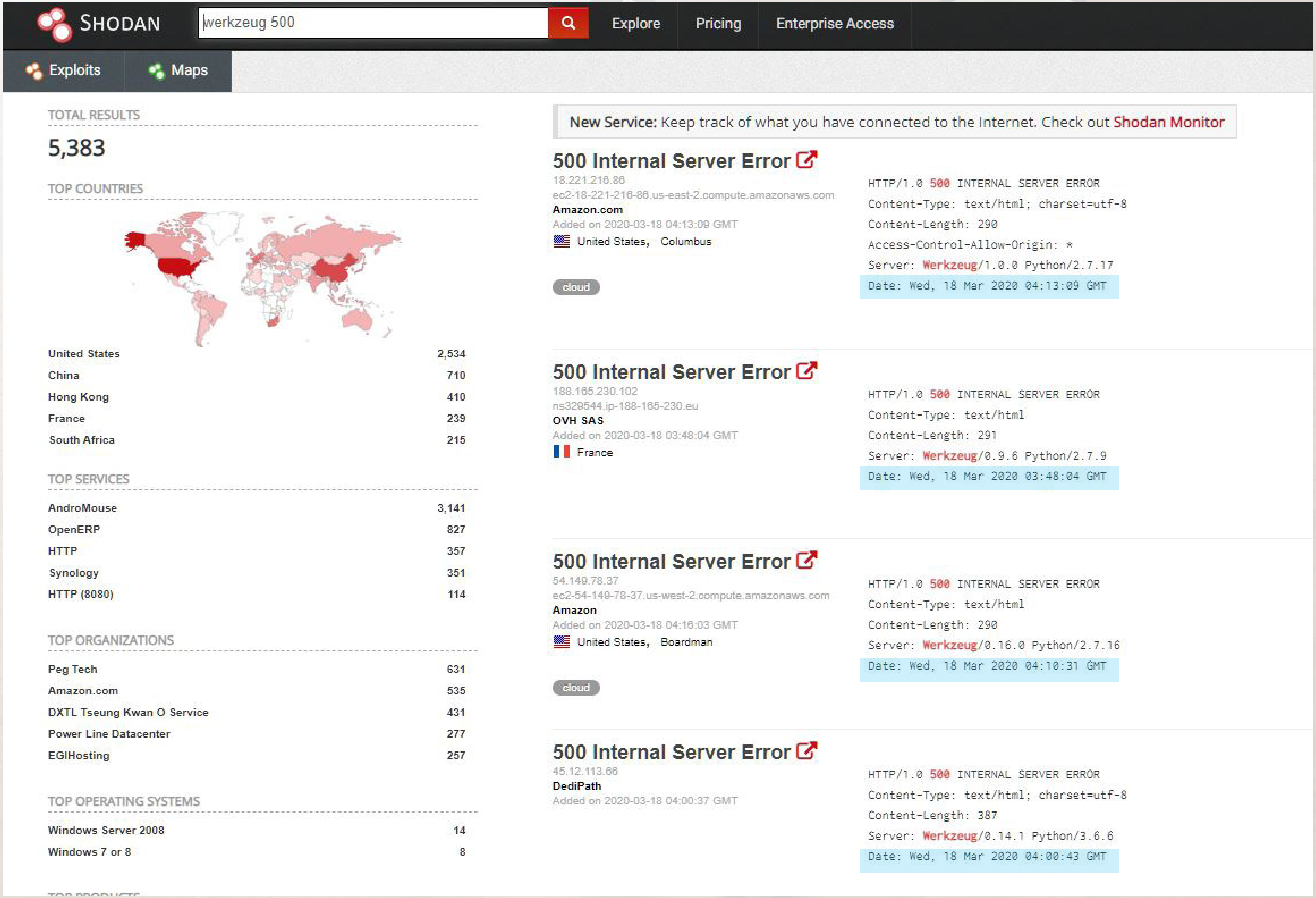
SSLv3, TLSv1, TLSv1.1, TLSv1.2

Diffie-Hellman Parameters

Fingerprint: RFC3526/Oakley Group 14

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Date: Sat, 05 Sep 2015 11:30:25 GMT
Server: Werkzeug/0.9.6 Python/3.4.0
Content-Type: text/html; charset=utf-8
X-XSS-Protection: 0
Connection: close
Transfer-Encoding: chunked
```

"BUT THAT WAS FROM 2015" YOU MIGHT SAY...



TOTAL RESULTS

5,383

TOP COUNTRIES

United States	2,534
China	710
Hong Kong	410
France	239
South Africa	215

TOP SERVICES

AndroMouse	3,141
OpenERP	827
HTTP	357
Synology	351
HTTP (8080)	114

TOP ORGANIZATIONS

Peg Tech	631
Amazon.com	535
DXTL Tseung Kwan O Service	431
Power Line Datacenter	277
EGIHosting	257

TOP OPERATING SYSTEMS

Windows Server 2008	14
Windows 7 or 8	8

New Service: Keep track of what you have connected to the Internet. Check out Shodan Monitor

500 Internal Server Error

18.221.216.86
ec2-18-221-216-86.us-east-2.compute.amazonaws.com
Amazon.com
Added on 2020-03-18 04:13:09 GMT
United States, Columbus

cloud

```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 290
Access-Control-Allow-Origin: *
Server: Werkzeug/1.0.0 Python/2.7.17
Date: Wed, 18 Mar 2020 04:13:09 GMT
```

500 Internal Server Error

188.165.230.102
ns329544.ip-188-165-230.eu
OVH SAS
Added on 2020-03-18 03:48:04 GMT
France

```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html
Content-Length: 291
Server: Werkzeug/0.9.6 Python/2.7.9
Date: Wed, 18 Mar 2020 03:48:04 GMT
```

500 Internal Server Error

54.149.78.37
ec2-54-149-78-37.us-west-2.compute.amazonaws.com
Amazon
Added on 2020-03-18 04:16:03 GMT
United States, Boardman

cloud

```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html
Content-Length: 290
Server: Werkzeug/0.16.0 Python/2.7.16
Date: Wed, 18 Mar 2020 04:10:31 GMT
```

500 Internal Server Error

45.12.113.66
DediPath
Added on 2020-03-18 04:00:37 GMT

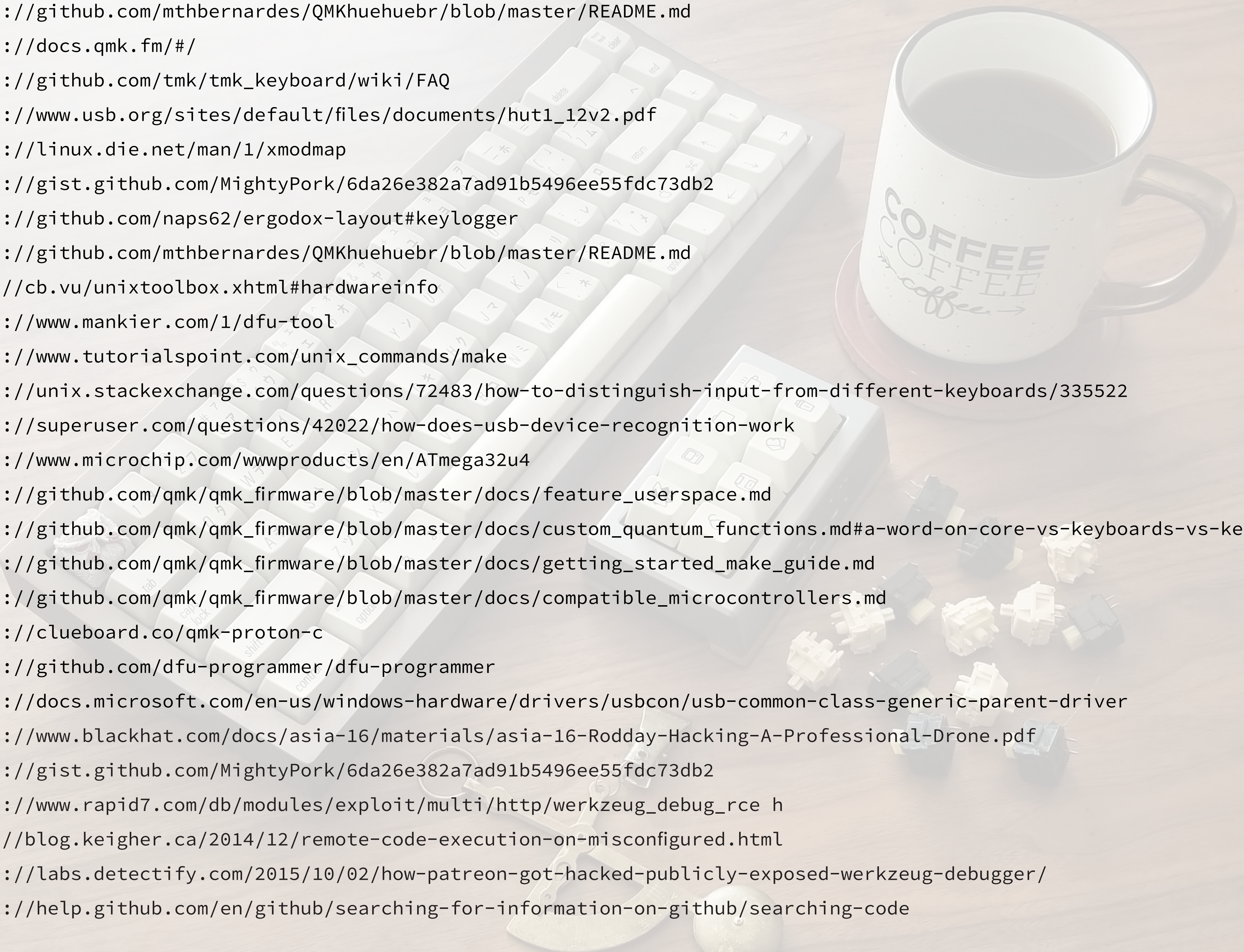
```
HTTP/1.0 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 387
Server: Werkzeug/0.14.1 Python/3.6.6
Date: Wed, 18 Mar 2020 04:00:43 GMT
```

SO WHAT DO YOU LOOK FOR?

- Debug / debugger / debug mode
- Developer mode / features
- Unsecured APIs
- App Suite Features (Microsoft Office for example)
- Troubleshoot methods / capabilities
- ‘Test’ cases

Special thanks to /u/superuser41 for all the help with putting this together.

Sources:



https://github.com/mthbernardes/QMKhuehuebr/blob/master/README.md

https://docs.qmk.fm/#/

https://github.com/tmk/tmk_keyboard/wiki/FAQ

https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf

https://linux.die.net/man/1/xmodmap

https://gist.github.com/MightyPork/6da26e382a7ad91b5496ee55fdc73db2

https://github.com/naps62/ergodox-layout#keylogger

https://github.com/mthbernardes/QMKhuehuebr/blob/master/README.md

http://cb.vu/unixtoolbox.xhtml#hardwareinfo

https://www.mankier.com/1/dfu-tool

https://www.tutorialspoint.com/unix_commands/make

https://unix.stackexchange.com/questions/72483/how-to-distinguish-input-from-different-keyboards/335522

https://superuser.com/questions/42022/how-does-usb-device-recognition-work

https://www.microchip.com/wwwproducts/en/ATmega32u4

https://github.com/qmk/qmk_firmware/blob/master/docs/feature_userspace.md

https://github.com/qmk/qmk_firmware/blob/master/docs/custom_quantum_functions.md#a-word-on-core-vs-keyboards-vs-keymap

https://github.com/qmk/qmk_firmware/blob/master/docs/getting_started_make_guide.md

https://github.com/qmk/qmk_firmware/blob/master/docs/compatible_microcontrollers.md

https://clueboard.co/qmk-proton-c

https://github.com/dfu-programmer/dfu-programmer

https://docs.microsoft.com/en-us/windows-hardware/drivers/usbcon/usb-common-class-generic-parent-driver

https://www.blackhat.com/docs/asia-16/materials/asia-16-Rodday-Hacking-A-Professional-Drone.pdf

https://gist.github.com/MightyPork/6da26e382a7ad91b5496ee55fdc73db2

https://www.rapid7.com/db/modules/exploit/multi/http/werkzeug_debug_rce h

ttps://blog.keigher.ca/2014/12/remote-code-execution-on-misconfigured.html

https://labs.detectify.com/2015/10/02/how-patreon-got-hacked-publicly-exposed-werkzeug-debugger/

https://help.github.com/en/github/searching-for-information-on-github/searching-code