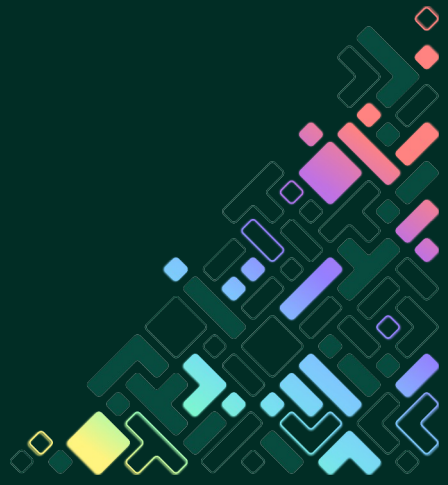
Zephyr® Project
Developer Summit

Flexboard

Developing Zephyr on a Keyboard that Runs it

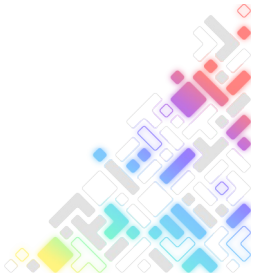
Daniel DeGrasse

Intro



Who Am I?

- Embedded Engineer at NXP
- Maintainer of Disk and MIPI DBI Subsystems
- Currently focused on displays and video applications.
- Otherwise, generally active in Zephyr upstream for NXP enablement

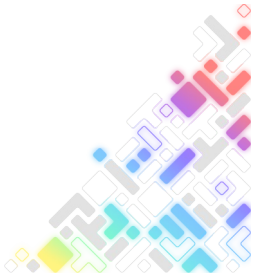


Why Build A Keyboard?

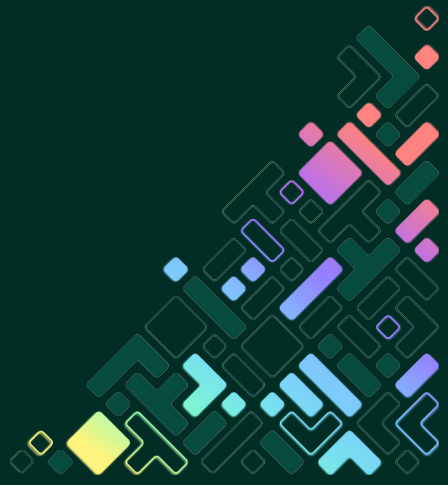
- Fun Challenge
- Product Design without the pressure
- Specific Design Requirements
  - 145/110 Key build
  - Individual under key backlighting

Component Selection Process

- Began during chip shortage
- In tree SOC (Kinetis K22)
  - USB support, 120MHz core clock
  - High pin count
- LED controller needed to drive LED matrix
  - No existing support for this controller in tree

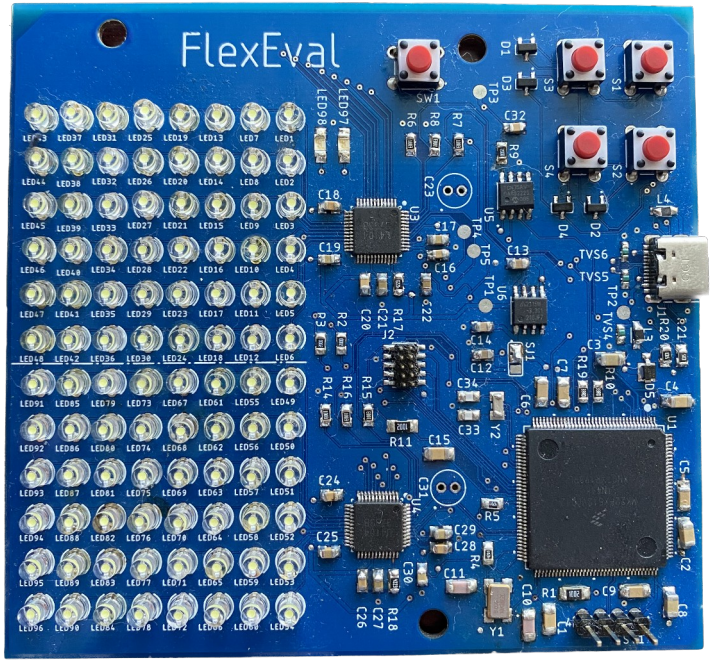


PCB and Case Design



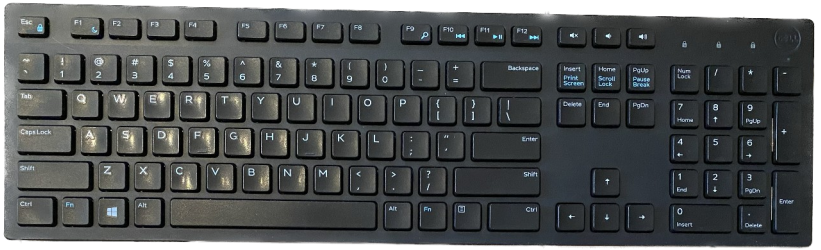
FlexEval

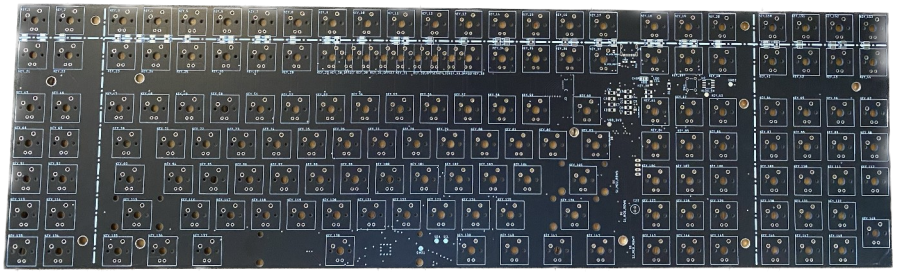
- Simple design to validate MCU functionality and LED controller
- Includes dual LED controllers
- Added a TCN75 temp sensor for future use
- Also useful as a development platform for new ZMK features



FlexBoard

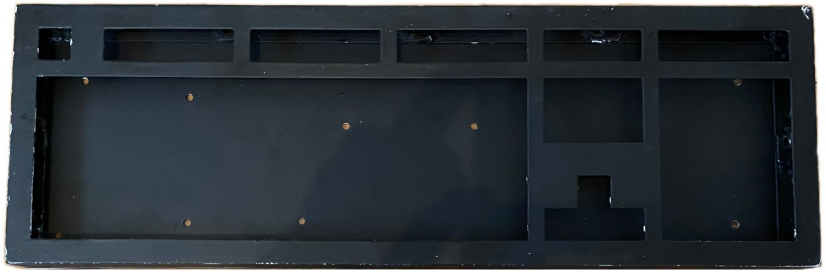
- 145 Key Matrix
- Single LED controller for entire array
- Designed PCB to be cut to enable multiple sizes
- Included FFC footprint for potential expansion



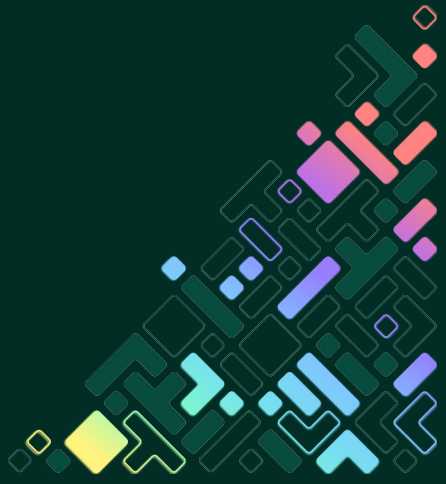


Case Design

- Alloy steel switch plate
- Wooden base and Aluminum siding
- Full sized case designed entirely from steel

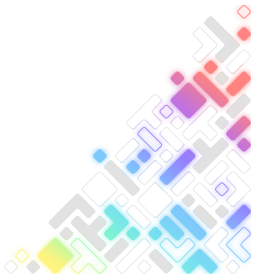


Porting Zephyr



Bringup Process

- Setup out of tree Board definition and manifest repository with west
- Added basic board definition with UART and GPIO pins
- Setup a loop during early init, to avoid bricking the SOC if firmware had issues
- Immediately hit a build error at the SOC level
- ???



Bringup Process- Take 2

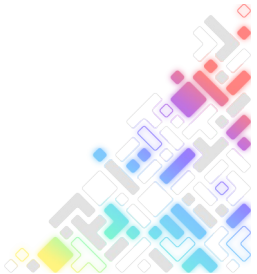
- Turns out this skew of the K22 is *not* the same as the K22 in tree
- First clue- slightly different clock control registers
- Turns out we’re doing a (relatively basic) SOC port

```
/** OSC - Register Layout Typedef */
typedef struct {
  __IO uint8_t CR;
} OSC_Type;
```

```
/** OSC - Register Layout Typedef */
typedef struct {
  __IO uint8_t CR;
       uint8_t RESERVED_0[1];
  __IO uint8_t DIV;
} OSC_Type;
```

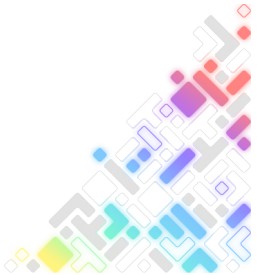
Bringup Process- SOC Port

- Out of tree SOCs are well supported in Zephyr, but *almost* all of this SOC support is already in tree
- Options:
  - Fork Zephyr, patch in tree support
  - Duplicate SOC out of tree
- NXP HAL also needed new register definitions for this part
  - No way to extend HAL, so I created a downstream fork
- At this point I opted to do bringup on downstream fork and upstream as I went
- Once in tree support was patched, Zephyr booted up and got UART output
- One more snag- USB doesn’t work
  - Turns out this part has NXP’s SYSMPU

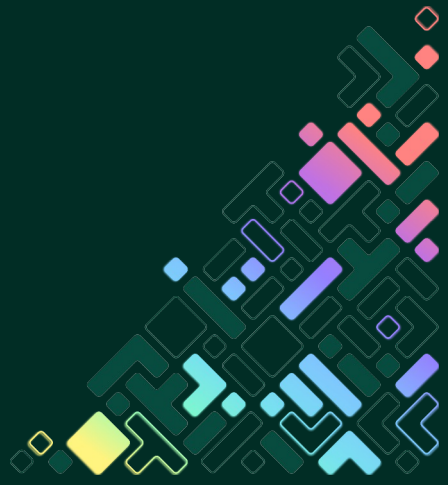


Driver Development

- Added driver for TCN75A temp sensor- upstreaming was painless
- Next, enabled IS31FL3733 LED controller
  - More difficulty here- controller supports “auto breathe mode”
  - This feature was not upstreamable, but rest of driver was merged
- Community response time on both PRs was excellent

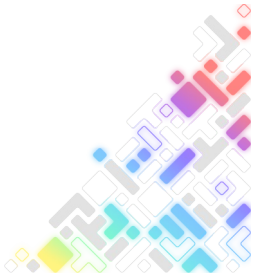


ZMK Support



Intro to ZMK

- Zephyr based keyboard framework
- MIT licensed
- USB HID and Bluetooth support
- Keyboard features supported via “behaviors”
  - Keymap Key macro RGB underglow, and Key layers
- Configured via devicetree settings
- Reconfiguration requires rebuilding image
  - Can be build locally, or via a github action



Initial Port

- ZMK requires additional devicetree definitions for key matrix and keymap
- ZMK_CONFIG variable set via west config cmake-args
  - directory for configuration overlays
- Keymap is set via bindings, which map to software “devices” for handling key events
- Keyscan worker will issue events to these devices, which will then queue key events
- Keycode event handler will consume key events and update HID reports to send to host

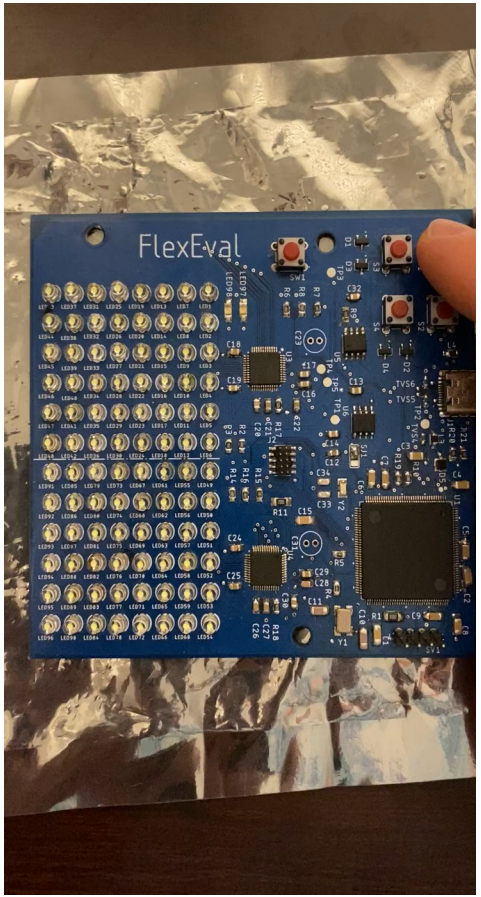
```
kscan0: kscan {
        compatible = "zmk,kscan-gpio-matrix";
        label = "KSCAN";
        diode-direction = "col2row";
        row-gpios = <&gpioe 2 (GPIO_ACTIVE_HIGH | GPIO_PULL_DOWN)>,
                <&gpioe 3 (GPIO_ACTIVE_HIGH | GPIO_PULL_DOWN)>;
        col-gpios = <&gpioe 8 GPIO_ACTIVE_HIGH>,
                <&gpioe 12 GPIO_ACTIVE_HIGH>;
};
```

```
keymap {
        compatible = "zmk,keymap";

        /* Default layer implements 1-3 key matrix, and function key */
        default_layer {
                bindings = <&kp A &kp NUMBER_1
                        &kp CAPS &mo FUNC>;
        };
```

Animation Support

- ZMK currently only supports RGB underglow upstream
- Needed support for per-key animations
- https://github.com/zmkfirmware/zmk/pull/1046
  - Already implemented in a PR
  - Extended with support for monochrome LEDs
- Needed a to make a downsteam fork of ZMK to use this feature
- Implementation uses Zephyr LED API- LED driver fit nicely into this PR

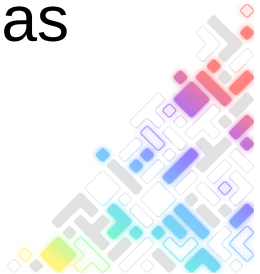


Keymap Configuration

- ZMK currently stores keymap in ROM
  - Move keymap data to RAM, load at boot time via NVS subsystem
- Keymap configuration performed via USB HID feature reports
  - Legacy Zephyr USB stack will provide callbacks for get/set feature reports
  - Used ZMK event subsystem to forward feature report events to settings subsystem
  - Settings subsystem responsible for configuring keys based on feature report contents
- Updated HID descriptor for keyboard to report additional endpoints for configuring keyboard
- Host side tooling- simple C program to send set/get feature reports to the keyboard to program keys

Downstream Fork Maintenance

- Rebased onto 3.3 after release
  - Carried patches to drivers and USB support forwards as needed
- ZMK patching was much heavier
  - Rebased animation PR onto latest ZMK, currently using a frozen branch
- Generally have not updated unless there is a feature needed from upstream- cherry picked patches back as needed

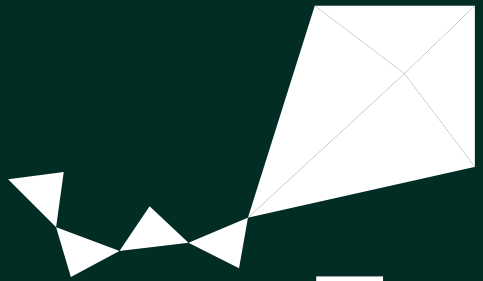


Final Product

- Runs custom ZMK firmware, built on top of Zephyr 3.3
- Firmware can be updated via MCUBoot USB DFU mode
- Support for keymap configuration via custom USB endpoints
- Per key backlight controls
- Full n-key rollover supported, along with media controls

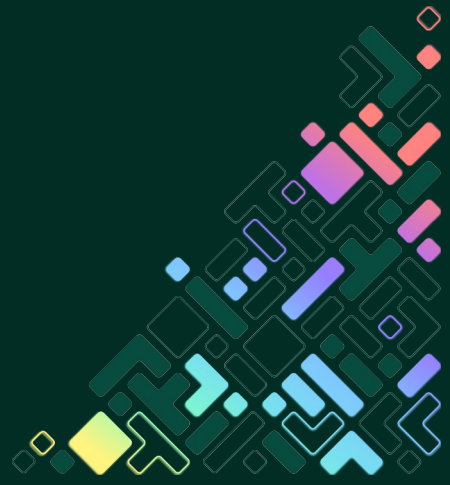
Conclusions

- May be value in exploring new methods to extend SOCs downstream
- HALs should consider extensibility options
- ZMK offers an interesting example of a downstream application
  - Slow to update to new Zephyr revisions- is Zephyr's change velocity too fast?
  - Uses a separate event subsystem- could probably be changed to ZBUS in the future
  - Active developer community- does seem that PRs occasionally get stalled in the review phase
- USB issues
  - No support for remote wakeup in Kinetis USB driver, USB DFU sample did not function out of box. Increased testing for regressions? Seems like USB frequently slips under radar

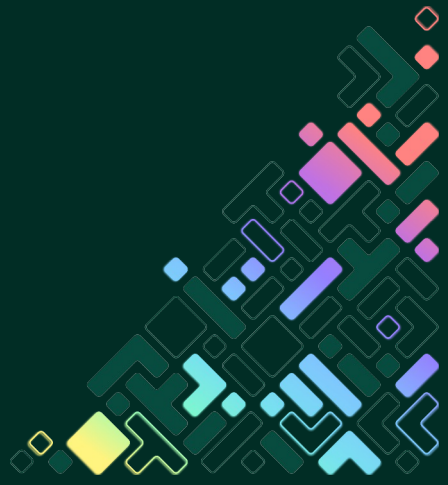


Zephyr® Project

Developer Summit

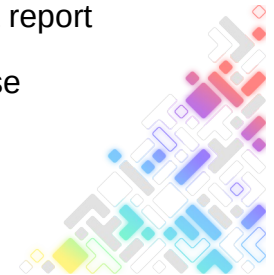


Backup Slides



HID Devices

- Describe the features of a USB device
- OS uses these to determine how to interact with USB device
- Descriptors need to be formatted right, or OS will ignore keyboard reports
- HID reports have 3 types (all generated by host):
  - Input: input events like keypresses
  - Output: only directed at device
  - Feature: reports generated by host, can set or get data from device
- ZMK describes HID endpoints for consumer keys (play/pause), standard keyboard reports (104 key keyboard), and modifier keys (alt/win)
- ZMK will set fields in input report when keys are pressed, and respond to periodic host request for input report
- Zephyr provides callbacks for output, input, and feature get/set reports, so ZMK simply implements these



Animation Framework

- Implemented via a series of “animation drivers”
  - Each driver provides APIs to enable/disable animation, and to update a rendered frame
- Animation subsystem runs at configurable FPS, will trigger a frame update at each frame interval
- Final frame is streamed via LED API to LED device
- Subsystem supports color blending, ripple animations, and solid colors
- Relations of pixel coordinates to key indices are defined in devicetree
- Animations can be limited to keyboard regions via zones
- Added custom animation to overlay LEDs for caps lock and num lock indicators