RAPID PROTOTYPING WITH ZEPHYR

Elektronikutvecklingsbyrån EUB AB

Tobias Olausson

$ WHOAMI

Short summary

- Cat person, but married to a human
- Dance Dance Revolution player
- Haskell enthusiast
- Embedded Software Engineer @ EUB

ZEPHYR HISTORY

Wind River

- 2001: Wind River buys Virtuoso RTOS
- 2015: Renames to Rocket and released as FOSS
- 2016: Renames to Zephyr, under Linux Foundation umbrella

Linux Foundation

- 2021: 250+ boards supported
- 2022: Most contributed RTOS out there
- 2023: 500+ boards supported

WHY ZEPHYR?

Community

- ❒ Free and open source software
- ❒ Permissive licensing
- ❒ Actively maintained

Features

- ❒ Large ecosystem (batteries included!)
- ❒ Hardware agnostic (device trees!)
- ❒ Configuration via KConfig
- ❒ Samples for almost everything

ZEPHYR COMPARED TO FREERTOS

Features

- Same primitives as FreeRTOS and more
  - Concurrency programming less based on threads
- Batteries included:
  - Logging subsystem
  - File system support and other NVS wrappers

Philosophy

- Opinionated design
- Dynamic memory allocation is not encouraged
- Decoupling of hardware and software
  - Run led sample on any board with a `led0`

DEMO

Blinky sample

```
$ mkdir workspace && cd workspace
$ git clone https://github.com/zephyrproject-rtos/zephyr.git
$ cd zephyr
$ west init .
$ west update
$ cd samples/basic/blinky/
$ west build -b nrf52840dk_nrf52840
$ west flash
```

DEMO

Blinky sample

```
$ mkdir workspace && cd workspace
$ git clone https://github.com/zephyrproject-rtos/zephyr.git
$ cd zephyr
$ west init .
$ west update
$ cd samples/basic/blinky/
$ west build -b nrf52840dk_nrf52840
$ west flash
```

On different board

```
$ rm -rf build
$ west build -b nrf5340dk_nrf5340_cpuapp
$ west flash
```

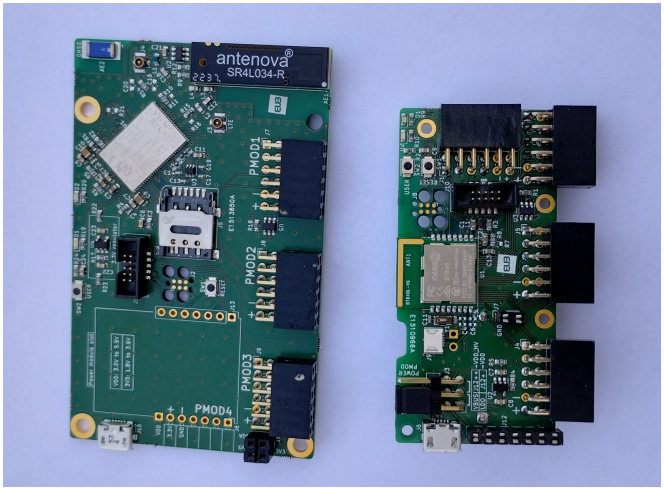

Figure 1: EUB Kite uses a Nordic SoC + PMOD

PMOD

- Open standard by Digilent
- Single row (6 pin) or double row (12 pin)
- SPI, I2C, UART, GPIO
- Just pinmux in the device tree

ZEPHYR SETUP

Hardware

- We always start from EUB Kite as DevKit
- Device tree overlay for specific peripherals
- Final HW dts just a merge of the kite and overlay

Software

- Using nRF connect Zephyr libraries
- Zephyr libs:
  - LED, UART, Watchdog, Events, Shell,
  - BSD Sockets, CoAP, DTLS

ZEPHYR AS AN RTOS

What you'd expect from any RTOS

- Semaphores
- Mutexes
- Threads
- etc

But also

- Work queues
- zbus
- ...and much more!

SEMAPHORES

Definition

```
/* Defined and initialized at compile-time */
K_SEM_DEFINE(name, initial_count, count_limit)
```

Example

```
/* Binary semaphore */
K_SEM_DEFINE(my_binary_sem, 0, 1);

/* Take the semaphore with timeout */
k_sem_take(&my_binary_sem, K_FOREVER);

/* Give the semaphore */
k_sem_give(&my_binary_sem);
```

THREADS

Definition

```
/* Define and initialize at compile-time. Starts automatically */
K_THREAD_DEFINE(name, stack_size, entry, p1, p2, p3, prio, options, delay)
```

Example

```
void eub_worker_thread(void)
{
    while (true) {
        ...
    }
}
K_THREAD_DEFINE(worker_thread, 4096, eub_worker_thread, NULL, NULL, NULL, 7, 0, 0);
```

ZEPHYR DRIVER MODEL

Devices trees...

- ...has the same syntax and semantics as in Linux
- ...specify the hardware setup
- ...annotates the spec with labels
- ...can be overlayed for small modifications
- ...are compiled and accessed with macros
  - ...and gives crazy error messages when wrong

EXAMPLE DEVICE TREE OVERLAY

Some LED and button setup

```
/ {
    leds {
        compatible = "gpio-leds";
        led2: led_2 {
            gpios = <&gpio0 7 (GPIO_ACTIVE_LOW | NRF_GPIO_DRIVE_HOH1)>;
            label = "Awake LED";
        };
    };

    buttons {
        button1: button_1 {
            gpios = <&pmod4 2 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>;
            label = "EXT_BUTTON";
        };
    };

    aliases {
        led2 = &led2;
        power-led = &led2;
        connection-good-led = &led0;
        connection-bad-led = &led1;
        external-button = &button0;
    };
};
```

DEVICE TREE COMPILATION

build/zephyr/include/generated/devicetree_generated.h

```
#define DT_N_S_leds_S_led_2_EXISTS 1
#define DT_N_ALIAS_led2      DT_N_S_leds_S_led_2
#define DT_N_ALIAS_power_led DT_N_S_leds_S_led_2
#define DT_N_NODELABEL_led2  DT_N_S_leds_S_led_2

#define DT_N_S_leds_S_led_2_PATH "/leds/led_2"
#define DT_N_S_leds_S_led_2_PARENT DT_N_S_leds
```

DEVICE TREE USAGE

The application knows nothing about where the LEDs are located, changing boards is possible without changing code!

Using the LEDs

```
const struct gpio_dt_spec connection_good_led =
        GPIO_DT_SPEC_GET(DT_ALIAS(connection_good_led), gpios);
const struct gpio_dt_spec connection_bad_led =
        GPIO_DT_SPEC_GET(DT_ALIAS(connection_bad_led), gpios);
const struct gpio_dt_spec power_led =
        GPIO_DT_SPEC_GET(DT_ALIAS(power_led), gpios);

void blink_led(const struct gpio_dt_spec *led, int times)
{
    for (int i = 0; i < times; i++) {
        gpio_pin_set_dt(led, 1);
        gpio_pin_set_dt(led, 0);
    }
}
```

Just like in Linux

- Board supplies a default config
- Applications overlay specifics in `prj.conf`
- Enables/disables drivers and entire subsystems
- Configs may enable other configs

KCONFIG

Just like in Linux

- Board supplies a default config
- Applications overlay specifics in `prj.conf`
- Enables/disables drivers and entire subsystems
- Configs may enable other configs

Example

```
CONFIG_MAIN_STACK_SIZE=4096
CONFIG_COAP=y
CONFIG_GPIO=y
CONFIG_NRF_MODEM_LIB=y
```

KCONFIG

Just like in Linux

- Board supplies a default config
- Applications overlay specifics in `prj.conf`
- Enables/disables drivers and entire subsystems
- Configs may enable other configs

Example

```
CONFIG_MAIN_STACK_SIZE=4096
CONFIG_COAP=y
CONFIG_GPIO=y
CONFIG_NRF_MODEM_LIB=y
```

Debugging

Merged KConfig with all dependencies in `build/zephyr/.config`

Sample manifest

```
manifest:
  self:
    path: device-firmware

  remotes:
    - name: ncs
      url-base: https://github.com/nrfconnect

  projects:
    - name: nrf
      repo-path: sdk-nrf
      remote: ncs
      revision: v2.2.0
      import:
         name-allowlist:
          - mcuboot
          - mbedtls-nrf
          - nrfxlib
          - zephyr
          - hal_nordic
          - segger
```

WEST - MANIFEST FOR ZEPHYR PARTS

Install west by following

...the Zephyr getting started guide

Get started with west

```
$ mkdir ~/my-project
$ cd ~/my-project
$ git clone <firmware-uri> device-firmware
$ cd device-firmware
$ west init -l .
$ west update
```

WEST IS EXTENDED BY ZEPHYR

Building

```
# clean build
$ west build -p -b <board_name>
# rebuild
$ west build
```

Flashing

```
$ west flash
```

Debugging

```
$ west attach
# flash and attach
$ west debug
```

NORDIC MODEM SUPPORT

No more manual AT commands! PPP implementation can be used for modems that are non-nordic.

Connecting

```c
#include <modem/nrf_modem_lib.h>
#include <modem/lte_lc.h>

err = lte_lc_init();
err = lte_lc_connect(void);
if (err) {
    /* Something went wrong */
}

/* Otherwise, we're now connected */
```

Debugging

- Modem tracing helps a LOT in debugging

LINKS

Zephyr resources

- https://github.com/zephyrproject-rtos/zephyr/
- https://docs.zephyrproject.org/latest/
- https://docs.zephyrproject.org/latest/kconfig.html

Nordic

- https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.4.1/nrf/index.html