

# Lab 07: Zephyr Devicetree, GPIO & Callbacks

## Medical Electrical Equipment (BME590L)

2023-03-27

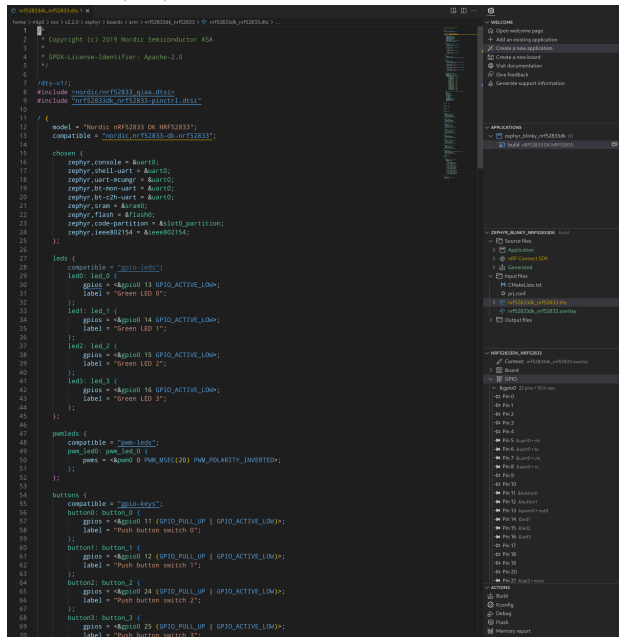
This week we will start programming the firmware that will be controlling your nRF52833DK.

## 1 Application Setup

1. Create a New Application based on `zephyr/samples/basic/blink`. This will create a new directory that will contain all of the necessary project files.
2. If you are comfortable with `git`, please `git init` this directory and associate it with an online repository on either GitHub or GitLab (Duke).<sup>1</sup>

## 2 Devicetree

1. Inspect the default Devicetree (DT) for the nRF52833DK.



```
1 // SPDX-License-Identifier: Apache-2.0
2 // Copyright (c) 2018 Nordic Semiconductor ASA
3 //
4 // #include <devicetree/bindings/gpio/nrf52833-pinmux.dtsi>
5 // #include <nrf52833-pinmux.dtsi>
6 //
7 #if !defined(CONFIG_GPIO)
8 #error "GPIO is not enabled in the kernel configuration."
9 #endif
10
11 / {
12     model = "nordic,nrf52833-dk-nrf52833";
13     compatible = "nordic,nrf52833-dk,nrf52833";
14
15     chosen {
16         zephyr_console = &uart0;
17         zephyr_shell_uart = &uart0;
18         zephyr_uart_rcmgr = &uart0;
19         zephyr_nrf_uart = &uart0;
20         zephyr_bt_uart = &uart0;
21         zephyr_flash = &flash0;
22         zephyr_code_partition = &slot0_partition;
23         zephyr_reset_gpio = &gpio0;
24     };
25
26     leds {
27         compatible = "gpio-leds";
28         led0: led_0 {
29             gpios = &gpio0 13 GPIO_ACTIVE_LOW;
30             label = "green LED 0";
31         };
32         led1: led_1 {
33             gpios = &gpio0 14 GPIO_ACTIVE_LOW;
34             label = "green LED 1";
35         };
36         led2: led_2 {
37             gpios = &gpio0 15 GPIO_ACTIVE_LOW;
38             label = "green LED 2";
39         };
40         led3: led_3 {
41             gpios = &gpio0 16 GPIO_ACTIVE_LOW;
42             label = "green LED 3";
43         };
44     };
45
46     pwmleds {
47         compatible = "pwm-leds";
48         pwm_led0: pwm_led_0 {
49             pwm = &pwm0 0 PWM_BSC(20) PWM_POLARITY_INVERTED;
50         };
51     };
52
53     buttons {
54         compatible = "gpio-keys";
55         button0: button_0 {
56             gpios = &gpio0 11 GPIO_PULL_UP | GPIO_ACTIVE_LOW;
57             label = "push button switch 0";
58         };
59         button1: button_1 {
60             gpios = &gpio0 12 GPIO_PULL_UP | GPIO_ACTIVE_LOW;
61             label = "push button switch 1";
62         };
63         button2: button_2 {
64             gpios = &gpio0 24 GPIO_PULL_UP | GPIO_ACTIVE_LOW;
65             label = "push button switch 2";
66         };
67         button3: button_3 {
68             gpios = &gpio0 25 GPIO_PULL_UP | GPIO_ACTIVE_LOW;
69             label = "push button switch 3";
70         };
71     };
72 }
```

<sup>1</sup>We will cover `git` in a few weeks if you are not familiar with it.

2. The nRF52833DK has 4 integrated LEDs. Create a DT overlay file for your project that creates alias for 5 LEDs on the following pins (4 of which are the integrated LEDs):

| Firmware Variable Name | DT Alias  | GPIO Pin |
|------------------------|-----------|----------|
| heartbeat_led          | heartbeat | P0.13    |
| buzzer_led             | buzzer    | P0.14    |
| ivdrip_led             | ivdrip    | P0.15    |
| alarm_led              | alarm     | P0.16    |
| error_led              | error     | P1.3     |

3. The nRF52833DK also has 4 integrated buttons. Create a DT overlay file for your project that creates aliases for these 4 buttons as follows:

| Firmware Variable Name | DT Alias | GPIO Pin    |
|------------------------|----------|-------------|
| sleep                  | button0  | You Tell Me |
| freq_up                | button1  | You Tell Me |
| freq_down              | button2  | You Tell Me |
| reset                  | button3  | You Tell Me |

### 3 Firmware

Implement the following firmware functionality:

- Check that both the GPIO0 and GPIO1 interfaces are ready.
- Initialize all LED output pins as GPIO\_ACTIVE\_LOW.
- Be sure to capture all function exit codes and have conditional statements to capture any returned error codes.
- Implement the following control logic:
  - The heartbeat LED blinks at a fixed 1 Hz while main() is being executed.
  - The 3 "action" LEDs (buzzer\_led, ivdrip\_led, and alarm\_led) cycle through each one being on for 1 s in that order (i.e., 3 s period for each to be on 1 s). This 1 s default on-time should be defined using a preprocessor macro: #define LED\_ON\_TIME\_S 1.
  - The freq\_up reduces the on-time of each "action" LED by 0.1 s (and, therefore, reduces the overall cycle period by 0.3 s) each time it is pressed. This incremental decrease in on-time should be defined by the preprocessor macro: #define DEC\_ON\_TIME\_S 0.1
  - The freq\_down increases the on-time of each "action" LED by 0.1 s (and, therefore, increases the overall cycle period by 0.3 s) each time it is pressed. This incremental increase in on-time should be defined by the preprocessor macro: #define INC\_ON\_TIME\_S 0.1
  - If the on-time for each individual "action" LED is < 0.1 s or > 2 s, then none of the action LEDs are illuminated and the error LED is continuously illuminated. Define each of this min/max limits using preprocessor macros (you can choose appropriate names).
  - If the error LED is illuminated, there is no response to pressing the freq\_up or freq\_down buttons until the reset button is pressed.
  - Pressing the reset button resets the on-time for each "action" LED to 1 s.
  - At any point in time, if the sleep button is pressed, all LEDs, except for the heartbeat LED, go off until the sleep button is pressed again, at which time the device returns back to the exact state it was in before being put to sleep.

## 4 Answer These Questions

1. What is the difference in configuring a GPIO LED pin to be `GPIO_ACTIVE_LOW` vs `GPIO_ACTIVE_HIGH`?
2. What is the purpose of calling `device_is_ready()` for the GPIO interfaces?
3. What GPIO pins are used for `button[0-3]`?
4. Why do we use callbacks to detect the button presses (instead of simply reading the `LOW/HIGH` state of each button GPIO pin in `main()`)?

## 5 Nordic Dev Academy

To get ready for next week's tasks, please complete the following lesson on the Nordic Dev Academy: nRF Connect SDK Fundamentals course:<sup>2</sup>

- Lesson 4 – Printing messages to console and logging
- Note - we will not use `print()`; everything will be done with the logging module.

## 6 What to Submit & Grading

**This lab exercise is due Monday, February 27 at 08:30.**

- Upload a PDF to Gradescope with answers to the questions posed above.
- If you used a git repository for your code, please include a URL in your PDF for that repository, and make sure that Dr. Palmeri has access to it.<sup>3</sup>
- If you didn't use a git repository, please upload a zip archive of your entire project to your Sakai Drop Box.
- Include in your PDF a screenshot showing completion of Lessons 1-4 from the Nordic Dev Academy.
- Code will be graded on functionality **and** efficiency of code logic **and** code "readability". "Readability" does not mean a lot of verbose comments; it means that the structure of the code, the naming of variables, etc. convey meaning and logical flow.

---

<sup>2</sup>Be sure to login and record your progress and quizzes to earn a certificate at the end of the course.

<sup>3</sup>Dr. Palmeri's username on both platforms is `m1p6`.