Table of Contents

1. Laboratory on GPIO use in Zephyr OS 2
1.1. Concept of the Device Tree 2
1.2. Use of the device defined in the device tree 4

1. Laboratory on GPIO use in Zephyr OS

GPIOs in Nordic Semiconductor nRF52840 can be handled by software with polling, with interrupts or using direct hardware connection with PPI (Programmable Peripheral Interconnect). In the first part of the laboratory, we will handle GPIO events with the software. The first example named “blinky” [1] is a simple program that blinks the onboard LED. It can be used to check if all settings are set properly, and if all hardware and software modules work. After compilation and uploading of the program one LED on the board (LED0) should blink. Using this example we can analyze the way of setting parameters and using the GPIO input ports in Zephyr RTOS.

1.1. Concept of the Device Tree

In Zephyr RTOS the device tree [2] concept is used to describe the hardware modules and their properties. The device tree file (*.dts) is specific for every board. Another file (*.dtsi) describes in detail the hardware peripherals built into the specific microchip. It contains names (labels) and addresses of registers of all peripherals in the SoC.

Let's focus on the device tree entry describing the LED connected to some GPIO output. This entry exists in the file that describes the board. In our case, it's the file for nRF52840DK located at the path:

```
C:\...\.platformio\packages\frameworkzephyr\boards\arm\nrf52840dk_nrf52840\nrf52840dk_nrf52840.dts
```

In the file, we can find the description of four LEDs. Let's show here one of them:

```
leds {
        compatible = "gpio-leds";
        led0: led_0 {
            gpios = <&gpio0 13 GPIO_ACTIVE_LOW>;
            label = "Green LED 0";
        };
             /*... details of other LEDs are here */
      };
```

The **compatible** key specifies compatibility with the hardware device. During the build system tries to bind the entry with the *.yaml file that describes the hardware from a specific vendor. In our example, we use a generic class of LED devices which do not have a specific vendor **gpio-leds**.

- **led0** is the node label that we can use to access the details of the device.
- **led_0** is the child node of gpio-leds. Here it has two properties: gpios and label.
- **gpios** property is required for gpio-leds class. It specifies the gpio port (here gpio0), bit number (here 13), and flags that define the behaviour of the GPIO (here active low output).
- **label** property is optional and contains human understandable name of the device.

Similarly, the buttons are described in the device tree file:

```
buttons {
```

```
        compatible = "gpio-keys";
        button0: button_0 {
            gpios = <&gpio0 11 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>;
            label = "Push button switch 0";
        };
             /*... details of other buttons are here */
        };
```

The generic class of button devices is **gpio-keys**. Other options are similar to LED but the flags. For the button, they additionally enable a pull-up resistor with GPIO_PULL_UP value. For better compatibility, it is possible to define the aliases of the entries. In the *.dts file there are aliases defined for leds and buttons.

```
/* These aliases are provided for compatibility with samples */
aliases {
    led0 = &led0;
    led1 = &led1;
    ...
    sw0 = &button0;
    sw1 = &button1;
    sw2 = &button2;
    sw3 = &button3;
};
```

Notice that the description in the *.dts file does not specify the details of the internal peripheral built into SoC e.g. the addresses of the registers. Such information can be found in the *.dtsi file which holds the description of the chip itself. The file is located in the path:

```
C:\...\.platformio\packages\framework-zephyr\dts\arm\nordic\nrf52840.dtsi
```

The entry for gpio0 which is referred from leds and buttons entries looks like this:

```
        gpio0: gpio@50000000 {
            compatible = "nordic,nrf-gpio";
            gpio-controller;
            reg = <0x50000000 0x200
                   0x50000500 0x300>;
            #gpio-cells = <2>;
            label = "GPIO_0";
            status = "disabled";
            port = <0>;
        };
```

Bindings of the entries are defined in YAML files. The flags property is defined as one of the #gpio-cells in the file:

```
C:\...\.platformio\packages\framework-zephyr\dts\bindings\gpio\nordic,nrf-gpio.yaml
```

as one of the entries:

```
gpio-cells:
  - pin
  - flags
```

1.2. Use of the device defined in the device tree

To use the hardware device in the program first there is a need to check if this device is defined in the device tree, and if so, set its main parameters. We can use the blinky example

1)

https://github.com/zephyrproject-rtos/zephyr/blob/main/samples/basic/blinky/src/main.c
2)

https://docs.zephyrproject.org/latest/build/dts/index.html

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - **RAU6 Dokuwiki**

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:s2:aih:lab:nrf52gpio?rev=1700647635

Last update: **2023/11/22 11:07**