SC620 Driver

Devicetree

Devicetree binding file

Devicetree binding describes how the device should be described in the devicetree (i.e. its properties and their types). For our SC620 driver two files were added:

- `dts/bindings/rg-led.yaml` : describes the properties of a single RG LED (Red-Green), which is defined as a child-binding and has the properties `idx_green` and `idx_red` indicating the index on the led controller to which the green and red leds are connected respectively.
- `dts/bindings/st,sc620.yaml` : describes the sc620 device. It includes `i2c-device.yaml` which makes it needs to be defined as a child of an i2c, as well as inheriting some properties related to i2c (i.e. reg). It also includes `rg-led.yaml` making that it can have multiple RG LED child properties.

Devicetree Node

Within the devicetree (`boards/arm/mvpi_pulse_STM32L071/mvpi_pulse_STM32L071.dts`), the sc620 is defined as below. It is linked to the binding file through the build system by the line `compatible: "st,sc620"`

```
&i2c1 {
    pinctrl-0 = <&i2c1_scl_pb6 &i2c1_sda_pb7>;
    pinctrl-names = "default";
    status = "okay";
    clock-frequency = <I2C_BITRATE_STANDARD>;

    ledcontroller: sc620@70 {
        compatible = "st,sc620";
        status = "okay";
        reg = <0x70>;
        enable-gpios = <&gpioh 0 GPIO_ACTIVE_HIGH>;

        led0: led_0 {
            idx_red= <3>;
            idx_green= <0>;
        };

        led1: led_1 {
            idx_red= <4>;
            idx_green= <5>;
        };
    };
};
```

The address of the sc620 is set to 0x70 and not 0xe0 (as found in the datasheet). This is due to I2C only defining the address as the first 7 bits and using the 8th bit to indicate R/W. The datasheet states the address as 8-bit (0xe0). Zephyr however wants only the first 7 bits of the address, hence we shift 0xe0 1 bit to the right and obtain 0x70.

Driver

The driver consists of 2 parts, a generic driver API and a concrete implementation. The idea behind this is to abstract the functionality (in our case a number of rg-leds to a led controller) from the actual implementation (i.e. the sc620 chip).

Driver API

located in `include/rg_led/rg_led.h`, this file defines the methods `rg_led_on, rg_led_off and rg_led_get_info` used to control the rg-leds. These methods all take a pointer to a device as their first argument and simply call the corresponding method on the `device->api` struct.

It also defines the `rg_led_api` struct itself, which should be present in the device as well as the `rg_led_info` struct.

SC620 driver

located in `drivers/rg-led/sc620.c`. It contains the concrete implementations for the `sc620_led_api` (which is of type `rg_led_api`). It also contains an initialization function which will:

- check if i2c device is ready
- check if gpio enable pin is available and configure it as an output

Last but not least, it contains the macro's which allow us to:

- obtain properties and configuration from the devicetree
- initialize an instance of the device containing the `sc620_led_api`, the configuration and initialization function. The device instance can then be retrieved in our main code by the `DEVICE_DT_GET` macro and leds are then set by passing this device to e.g. `rg_led_on`

The define `#define DT_DRV_COMPAT st_sc620` at the top of the file is what tells the build system that this is the driver related to the devicetree node with `compatible: "st,sc620"`

next the macro `#define RG_LED_INFO(led_node_id) ...` near the bottom of the file will construct a `rg_led_info` struct using the led_node_id and some macros to retrieve values from the devicetree.

the macro `#define SC620_DEFINE(id) ...` will construct the `sc620_config` struct using the id (which is the id of the sc620 in the device tree) alongside some macros to retrieve properties from the devicetee. The last line of this macro will create a device instance and couples the init functions, the configuration and the api to this instance.

Finally the `DT_INST_FOREACH_STATUS_OKAY(SC620_DEFINE)` will call SC620_DEFINE (i.e. create a device instance) for each sc620 device in the device tree.