Reset Button Code Implementation

Overview

The reset button state is readable from both U-Boot and the Linux kernel. It is tied to GPIO3_IO[19], which is mapped to a pin number via the formula

`(unit - 1) * 32 + num = (3 - 1) * 32 + 19 = 83`

In U-Boot, the button's GPIO value can be read from the U-Boot shell prompt (`=>`) via:

```
gpio input 83
```

The GPIO is pulled up, so its value is `0` when the reset button is being pressed, otherwise `1`.

Under Linux, the reset button state can be read via ioctl, e.g., using the `evtest` utility. From the command line:

```
evtest --query /dev/input/event1 EV_KEY BTN_0
echo $?
```

prints 10 when the button is pressed, otherwise 0.

GPIO Keys Device Tree Declaration

In U-Boot, a GPIO Keys device is declared in the device tree file *uboot/arch/arm/dts/imx7d-roadrunner.dtsi* and initialized in *uboot/board/revo/mx7d_roadrunner/mx7d_roadrunner.c*. The Linux kernel uses the same device tree file as U-Boot, located at *kernel/arch/arm/boot/dts/imx7d-roadrunner.dtsi* with no additional initialization.

GPIO Keys device declaration is as follows:

```
/ {
    ...
    gpio-keys {
        compatible = "gpio-keys";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_gpio_keys>;

        btn0 {
            label = "BTN0";
            gpios = <&gpio3 19 GPIO_ACTIVE_LOW>;
            linux,code = <BTN_0>;
            wakeup-source;
        };
    };
    ...
```

```
};

...
&iomuxc {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_hog_1>;

    imx7d-sdb {
        ...
        pinctrl_gpio_keys: gpio_keysgrp {
            fsl,pins = <
                MX7D_PAD_LCD_DATA14__GPIO3_IO19     0x32
            >;
        };
    };
...
};
```

The macro `MX7D_PAD_LCD_DATA14__GPIO3_IO19` is a tuple declared in the file *mx7d_pinfunc.h* under the DTS directory of both U-Boot and Linux - respectively, *uboot/arch/arm/include/asm/arch-mx7/mx7d_pinfunc.h* and *kernel/arch/arm/boot/dts/imx7d-pinfunc.h*. Though these files are not identical, their shared declarations are. The config value, `0x32`, maps according to the following table per kernel documentation *fsl,imx7d-pinctrl.txt*:

CONFIG Description[1]	CONFIG Bit Field
PAD_CTL_PUS_100K_DOWN	(0 << 5)
PAD_CTL_PUS_5K_UP	(1 << 5)
PAD_CTL_PUS_47K_UP	(2 << 5)
PAD_CTL_PUS_100K_UP	(3 << 5)
PAD_CTL_PUE	(1 << 4)
PAD_CTL_HYS	(1 << 3)
PAD_CTL_SRE_SLOW	(1 << 2)
PAD_CTL_SRE_FAST	(0 << 2)
PAD_CTL_DSE_X1	(0 << 0)
PAD_CTL_DSE_X2	(1 << 0)
PAD_CTL_DSE_X3	(2 << 0)
PAD_CTL_DSE_X4	(3 << 0)

[1] These are field descriptions only, not actual constants.

So `0x32` maps to

```
PAD_CTL_PUS_5K_UP | PAD_CTL_PUE | PAD_CTL_DSE_X3
```

See also i.MX7 Dual Reference Manual.

GPIO Keys Initialization

Confusingly, `MX7D_PAD_LCD_DATA14__GPIO3_IO19` is redefined in *uboot/arch/arm/include/asm/arch-mx7/mx7d_pins.h* as a 64-bit integer derived by ORing the tuple values. The 64-bit value is used to initialize the GPIO button in *uboot/board/revo/mx7d_roadrunner/mx7d_roadrunner.c* as follows:

```
#define GPIO_BUTTON     IMX_GPIO_NR(3, 19)

iomux_v3_cfg_t const reset_key_pads[] = {
    (MX7D_PAD_LCD_DATA14__GPIO3_IO19 | MUX_PAD_CTRL(BUTTON_PAD_CTRL)),
};

int is_reset_button_pressed(void)
{
    int button_pressed = 0;

    /* Check if reset button pressed */
    imx_iomux_v3_setup_multiple_pads(reset_key_pads,
        ARRAY_SIZE(reset_key_pads));
    gpio_request(GPIO_BUTTON, "user_button");
    gpio_direction_input(GPIO_BUTTON);

    if (gpio_get_value(GPIO_BUTTON) == 0) { /* User button is low assert */

        button_pressed = 1;
        printf("Reset pressed\n");
    }

    return button_pressed;
}

int board_init(void)
{
    ...
    /* Configure reset button */
    imx_iomux_v3_setup_multiple_pads(
        reset_key_pads, ARRAY_SIZE(reset_key_pads));
    gpio_request(GPIO_BUTTON, "user_button");
    gpio_direction_input(GPIO_BUTTON);
    ...
}
```

Additional References

fsl,imx-pinctrl.txt
pinctrl-bindings.txt