How to control a GPIO in kernel space

How to control a GPIO in kernel space

Stable: 24.01.2020 - 09:46 / Revision: 24.01.2020 - 09:45

Contents

1 Purpose

This article gives an example of a driver that **controls GPIOs from kernel space**.
Sample source files are provided as examples: **kernel module** (driver), **device tree** and **Makefile**.
This example is available for STM32MP15_Evaluation_boards or STM32MP15_Discovery_kits

2 Code

2.1 Objective

Sample gpiolib usage code that makes an LED blink for 20 seconds.

2.2 Device tree

```
dummy_device {
   compatible = "st,dummy";
   status = "okay";
   greenled-gpios = <&gpioa 14 0>;
};
```

See GPIO_device_tree_configuration for more details of GPIO use in a device tree.

2.3 Kernel module code

```
#include <linux/module.h>
#include <linux/of_device.h>
#include <linux/kernel.h>
#include <linux/delay.h>
#include <linux/gpio/consumer.h>
#include <linux/platform_device.h>

struct gpio_desc *red, *green;

static int gpio_init_probe(struct platform_device *pdev)
{
   int i = 0;

   printk("GPIO example init\n");

   /* "greenled" label is matching the device tree declaration. OUT_LOW is the value
at init */
   green = devm_gpiod_get(&pdev->dev, "greenled", GPIOD_OUT_LOW);

   /* blink of the green led */
   while (i < 10)
   {
        ssleep(1);
        gpiod_set_value(green, 1);

        ssleep(1);
        gpiod_set_value(green, 0);

        i++;
   }

   return(0);
}

static int gpio_exit_remove(struct platform_device *pdev)
{
   printk("GPIO example exit\n");

   return(0);
}

/* this structure does the matching with the device tree */
/* if it does not match the compatible field of DT, nothing happens */
static struct of_device_id dummy_match[] = {
    {.compatible = "st,dummy"},
    {/* end node */}
};

static struct platform_driver dummy_driver = {
    .probe = gpio_init_probe,
    .remove = gpio_exit_remove,
    .driver = {
        .name = "dummy_driver",
                .owner = THIS_MODULE,
                .of_match_table = dummy_match,
    }
};

module_platform_driver(dummy_driver);
```

```
MODULE_AUTHOR("Bernard Puel");
MODULE_DESCRIPTION("Gpio example");
MODULE_LICENSE("GPL");
MODULE_ALIAS("platform:dummy_driver");
```

2.4 Kernel module build

See Adding_external_out-of-tree_Linux_kernel_modules for further information on module compilation.

2.5 Kernel module use

```
PC $> scp dummy_driver.ko root@<board ip address>:/lib/modules/
```

- Update dependency descriptions for loadable kernel modules, and synchronize the data on disk with memory

```
Board $> /sbin/depmod -a
Board $> sync
```

- Insert the kernel module example into the Linux kernel

```
Board $> modprobe dummy_driver
[18167.821725] dummy_driver: GPIO example init
```

Light-emitting diode

General-Purpose Input/Output (A realization of open ended transmission between devices on an embedded level. These pins available on a processor can be programmed to be used to either accept input or provide output to external devices depending on user desires and applications requirements.)

Device Tree