Platform Device Drivers

What to Expect?

- W's of Platform Device Drivers
- Registering a Platform Driver
- Registering a Platform Device
- Binding a platform driver to a device
- Platform resources and platform data
- Testing a simple platform driver

W's of Platform Device Drivers

- Provides a mechanism to notify the kernel of available hardware on the board
- Mechanism to add the devices to the device model of the kernel
- Used for non-discoverable devices
- Driver for the devices on the virtual 'platform' bus

Components of Platform Device Drivers

- Two components
  - Platform Driver
    - Set of operations done on the device
  - Platform Device
    - Information about the device
    - Deemed to be connected to a virtual 'platform' bus

Platform Bus Drivers Registration

- platform_device.h
- platform_driver structure defined as below:
  - struct platform_driver
    - int (*probe)(struct platform_device *);
    - int (*remove)(struct platform_device *);
    - void (*shutdown)(struct platform_device *);
    - int (*suspend)(struct platform_device *, pm_message_t state);
    - int (*resume)(struct platform_device *);
  - At minimum, probe() & remove needs to be supplied
- int platform_driver_register(struct platform_driver *)

Platform Device Registration

- Defined by board specific file
- platform_device.h
- platform_device structure defined as below:
  - const char *name
  - int id
  - struct resource *resource
  - const struct platform_device_id *id_entry
- int platform_device_register(struct platform_device *pdev)

Binding the Driver with Device

- Mechanism for bus code to attach a driver to device
  - id_table
    - struct platform_device_id
      - char name[PLATFORM_NAME_SIZE]
      - kernel_ulong_t driver_data
  - Name of driver, specified in the name field

Specifying the Resource Info

- For providing the information such as memory locations, IRQ numbers etc
- struct resource my_resource [] = {

```
    {
     .start = RESOURCE_START_ADDRESS,
     .end  = RESOURCE_END_ADDRESS,
     .flags = IORESOURCE_MEM
    }
 }
```

- struct platform_device my_device = {

```
     .name = DRIVER_NAME,
     .num_resources = ARRAY_SIZE(my_resource),
     .resources = my_resource,
 }
```

Platform Data

- Mechanism to pass the generic device specific information from Platform Device to the Platform Driver
- Example – Passing GPIO information

```
int gpio_led = 53;
struct platform_device led_device {
    name = DRIVER_NAME,
    .dev = {
        .platform_data = &gpio_led,
    }
}
```

Platform Driver With DTB

-
  ```
  const struct of_device_id gpio_led_dt[] = {
      { .compatible = "my-led", },
      { }
  };
  ```
- `of_property_read_u32(np, "led-number", &gpio_number);`
- `.of_match_table = of_match_ptr(gpio_led_dt);`

What all we Learnt?

- W's of Platform Device Drivers
- Registering a Platform Driver
- Registering a Platform Device
- Binding a platform driver to a device
- Platform resources and platform data
- Testing a simple platform driver

Any Queries?