Embedded Systems Programming

Linux GPIO & I2C Drivers
(Module 13)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014
Linux GPIO Driver

- A GPIO (General Purpose Input/Output) pin can be configured, set up its direction, and value if it is an output pin.
- A SoC chip may have several GPIO components:
  - Multiple “gpio chips”
- A global number in the integrated GPIO namespace, i.e., 0, 1, 2,...,n
- sysfs interface to user space

### Diagram

```
+----------------+            +----------------+            +--------------------+
| GPIO framework|     ↓       | Quark GIP       |     ↓       | Quark legacy GPIO   |
| (gpiolib.c)   |     ↓       | controller      |     ↓       | GPIO_SUS[5:0]       |
|                |  ↓          | GPIO[7:0]       |  ↓          | GPIO[9:8]           |
| CY8C9540A      |            | 40 GPIO pins in |            |                     |
|                |            | 6 ports         |            |                     |
```

Real-time Systems Lab, Computer Science and Engineering, ASU
A driver for each GPIO controller to provide
- methods to establish GPIO direction and to access GPIO values
- method to return the IRQ number associated to a given GPIO
- flag saying whether calls to its methods may sleep
- optional base number

In `intel_qrk_gip_gpio.c`

```c
/* The base GPIO number under GPIOLIB framework */
#define INTEL_QRK_GIP_GPIO_BASE 8
/* The default number of South-Cluster GPIO on Quark. */
#define INTEL_QRK_GIP_NGPIO 8
```

In `include/linux/gpio/driver.h`, "gpio_chip" is defined, including
- base: identifies the first GPIO number handled by this chip.
- ngpio: the number of GPIOs handled by this controller; the last GPIO handled is (base + ngpio - 1).
GPIO Driver Operation

- GPIO chip driver request to add “gpio_chip” to the platform
  
  ```c
  gc->base = pdata->gpio_base;
  gc->ngpio = NGPIO;
  ret = gpiochip_add(&dev->gpio_chip);
  ```

- `gpiolib.c` exports methods to work on GPIO pins
  - from GPIO # to find chip and to invoke the corresponding methods provided by the chip
    
    ```c
    gpio_request_one(LED1, GPIOF_OUT_INIT_LOW, "led1");
    gpio_desc desc1 = gpio_to_desc(LED1);
    gpio_set_value(desc1, data);
    ```
  - sysfs gpio interfaces, such as
    
    ```c
    gpiod_export, gpio_unexport, gpiod_set_value,
    gpio_direction_input
    ```
I2C Drivers in Linux

- A driver for I2C bus
  - adapter and algorithm drivers
  - manages I2C bus transactions
- Drivers for I2C devices
- A client has the device’s I2C bus address and a pointer to a driver which is attached with an adapter
- When a user program issues a file operation that needs an I2C transaction
  - i2C_transfer (i2C-core.c) to invoke adap_algo_master_xfer
  - command or data is in an msg array
  - the adapter issues reads/writes to hardware I/O addresses.

(https://i2c.wiki.kernel.org/index.php/Driver_Architecture)
I2C and SMBus

- In general, a system can have multiple I2C buses via different adapters and many I2C devices
- 2-wire synchronous serial buses
  - Master and slave, addressable
- I2C bus and SMBus
  - compatible with each other.
- Differences
  - Timeout (in SMBus, reset interfaces when clock is low for longer than 35ms))
  - Maximum clock speed: 100MHz(Smbus) but I2C bus has both 400kHz and 3.4MHz versions.
  - Logic level: 1: 3V in I2C and 2.1V in SMBus
  - General call and alert response.
Example of Accessing I2C/SMBus Devices

```
for each i2c device
struct i2c_client {
    unsigned short  flags;
    unsigned short  addr;
    char  name[I2C_NAME_SIZE];
    struct i2c_adapter * adapter;
};
```

```
struct i2c_driver * driver;
struct device  dev;
int  irq;
struct list_head  list;
struct completion  released;
};
```

```
// for each i2c device
struct i2c_client {
    unsigned short  flags;
    unsigned short  addr;
    char  name[I2C_NAME_SIZE];
    struct i2c_adapter * adapter;
};
```

```
struct i2c_driver * driver;
struct device  dev;
int  irq;
struct list_head  list;
struct completion  released;
};
```