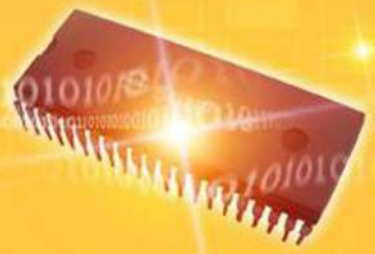
凌阳教育
www.sunplusedu.com
值得信赖的教育品牌

# 嵌入式系统工程师



Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

# IIC子系统

# 大纲

- i2c子系统概述
- i2c子系统组成
- 操作流程

# 大纲

- i2c子系统概述
- i2c子系统组成
- 操作流程

# i2c子系统概述

➢对比三种时序产生办法：

| 办法 | 特点 | 优缺点 |
| --- | --- | --- |
| 模拟I/O口 | 在对应时间节点把IO口拉高/低 | 思路清晰<br>操作麻烦 |
| 控制器 | 配置寄存器 | 操作麻烦<br>可移植性不好 |
| 子系统 | 内核把以上两种办法封装成函数接口 | 操作简单<br>可移植性好<br>初接触不好理解 |

# i2c子系统概述

- 两方面重点：

i2c子系统函数接口

有什么

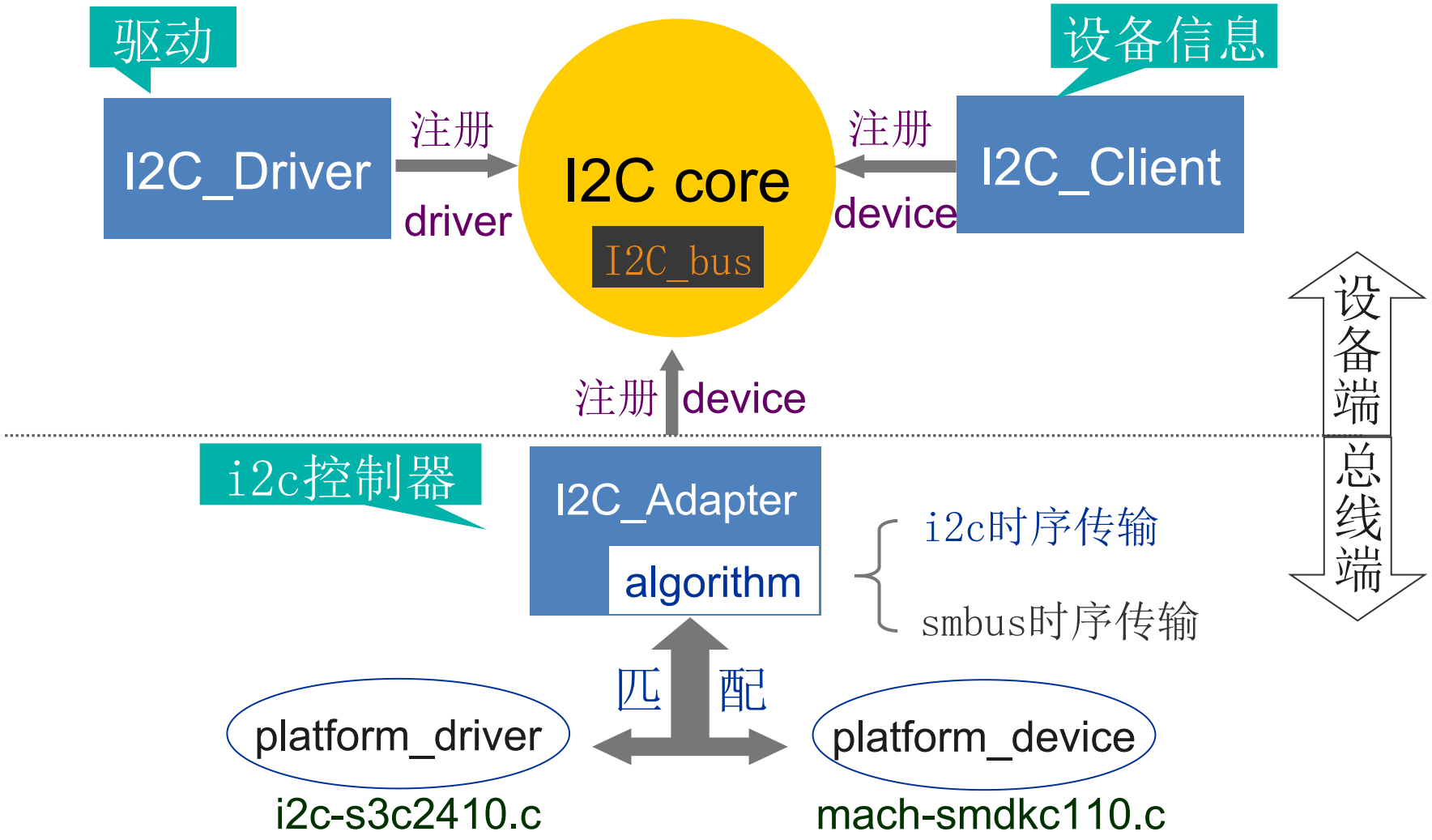
接口函数的使用流程

怎么用

# 大纲

- i2c子系统概述
- i2c子系统组成
- 操作流程

# i2c子系统组成


驱动
I2C_Driver
注册
driver
I2C core
I2C_bus
注册
device
I2C_Client
设备信息
注册
device
设备端
总线端
i2c控制器
I2C_Adapter
algorithm
i2c时序传输
smbus时序传输
匹配
platform_driver
i2c-s3c2410.c
platform_device
mach-smdkc110.c

# i2c子系统组成

## Core

子系统的核心。对适配器、设备及驱动进行管理。主要实现device和driver的注册、匹配、回调以及注销等操作。

## adapter

每个i2c控制器被抽象成了一个i2c_adapter,完成数据收任务。

## client

I2c设备被抽象成了一个i2c_client，用来描述设备资源。

## driver

完成i2c设备的控制功能,被抽象成i2c_driver，再将i2c设备注册成具体的字符、块或网络设备类型。

# i2c子系统组成

## ➢ 相关数据结构

struct i2c_algorithm *algo
传输协议算法

```
struct i2c_adapter {
        struct module *owner;
        unsigned int id;
        unsigned int class;              / * classes to allow probing for */
        const struct i2c_algorithm *algo; / * the algorithm to access the bus */
        void *algo_data;

        / * data fields that are valid for all devices      */
        struct rt_mutex bus_lock;

        int timeout;                     / * in jiffies */
        int retries;
        struct device dev;               / * the adapter device */

        int nr;
        char name[48];
        struct completion dev_released;

        struct list_head userspace_clients;
} ? end i2c_adapter ? ;
```

# i2c子系统组成

i2c时序产生办法

```
struct i2c_algorithm {

    int (*master_xfer)(struct i2c_adapter *adap, struct i2c_msg *msgs,
                int num);
    int (*smbus_xfer) (struct i2c_adapter *adap, u16 addr,
                unsigned short flags, char read_write,
                u8 command, int size, union i2c_smbus_data *data);

    /* To determine what the adapter supports */
    u32 (*functionality) (struct i2c_adapter *);
};
```

i2c_adapter：适配器

i2c_algorithm：一套通信方法，即一套时序。

具体设备通信就是用的i2c_algorithm指定的方法。

# i2c子系统组成

填充三要素：地址、名字和adapter

```
struct i2c_client {
        unsigned short flags;           / * div., see below          */
        unsigned short addr;            / * chip address -  NOTE: 7bit */
        char name[I2C_NAME_SIZE];
        struct i2c_adapter *adapter;    / * the adapter we sit on   */
        struct i2c_driver *driver;      / * and our access routines     */
        struct device dev;              / * the device structure        */
        int irq;                        / * irq issued by device        */
        struct list_head detected;
};
```

匹配后，由内核赋值

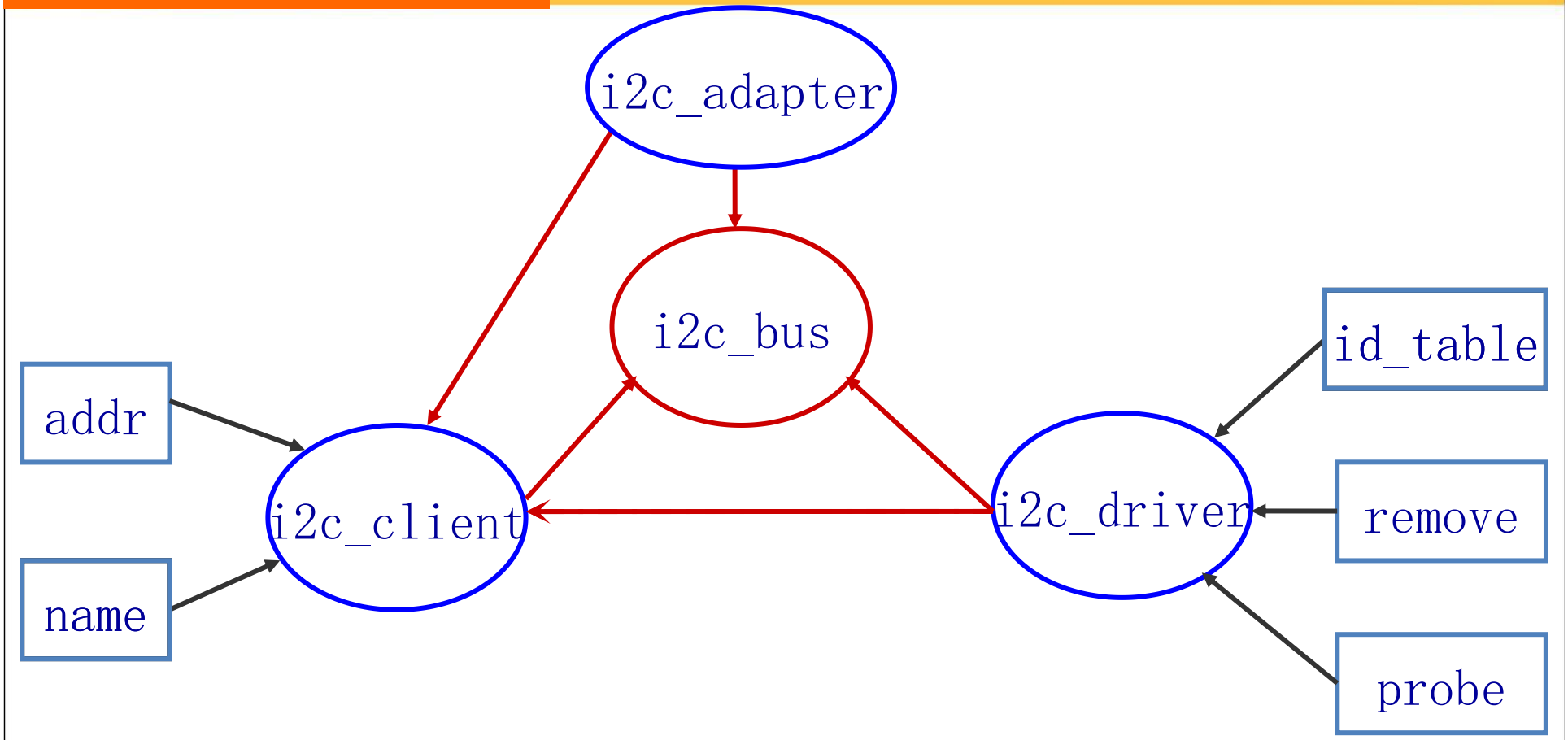
i2c_client描述了一个i2c设备，总线下有多少i2c设备就有多少个i2c_client与之对应。

# i2c子系统组成

```
struct i2c_driver {
        unsigned int class;
        int (*attach_adapter)(struct i2c_adapter *);
        int (*detach_adapter)(struct i2c_adapter *);

        / * Standard driver model interfaces */
        int (*probe)(struct i2c_client *, const struct i2c_device_id *);
        int (*remove)(struct i2c_client *);

        / * driver model interfaces that don't relate to enumeration  */
        void (*shutdown)(struct i2c_client *);
        int (*suspend)(struct i2c_client *, pm_message_t mesg);
        int (*resume)(struct i2c_client *);

        void (*alert)(struct i2c_client *, unsigned int data);

        / * a ioctl like command that can be used to perform specific functions
         * with the device.
         */
        int (*command)(struct i2c_client *client, unsigned int cmd, void *arg);

        struct device_driver driver;
        const struct i2c_device_id *id_table;
```

```
struct device_driver {
  const char  *name;
  struct module  *owner;
}
```

# i2c子系统组成



# 大纲

- i2c子系统概述
- i2c子系统组成
- 操作流程

# 操作流程

## ➢ 操作流程

- 设备端
  1. i2c_get_adapter
  2. i2c_new_device(相当于register设备)
  3. I2c_put_adapter
- 驱动端
  1. 填充i2c_driver
  2. i2c_add_driver(相当于register驱动)
  3. 在probe中建立访问方式

# 操作流程

## client相关函数

- 获得i2c_adapter结构体

  struct i2c_adapter *i2c_get_adapter(int id)

  参数：第几个adapter（0-2）

  返回值：获得的i2c_adapter结构体指针

- 创建并注册i2c_client

  struct i2c_client * i2c_new_device(struct i2c_adapter*adap,
  struct i2c_board_info const *info)

  参数1：i2c_adapter结构体指针

  参数2：i2c_boarid_info结构体指针（里面包含i2c设备地址，以及给这个设备起的名字）

  返回值：创建好并赋值的i2c_client结构体指针

# 操作流程

- 把i2c_client结构体从内核中删除

void i2c_unregister_device(struct i2c_client *client)

参数1：i2c_client 结构体指针

返回值：空

# 操作流程

## driver相关函数

- 把i2c_driver结构体加入内核

  int i2c_add_driver(srtuct i2c_driver *driver)

  参数1：i2c_driver结构体指针

  返回值：添加成功返回0，失败返回负值

- 把i2c_driver结构体从内核中删除

  void i2c_del_driver(struct i2c_driver *driver)

  参数1：i2c_driver结构体指针

  返回值：空

# 操作流程

- 数据传输办法
  - ❶结构体+统一函数
  - ❷单独的读/写函数
  - ❸smbus方式

---

①

填充一个或两个i2c_msg

int i2c_transfer(struct i2c_adapter *adap, struct i2c_msg *msgs, int num)

填充好的i2c_msg

i2c_msg的个数

# 操作流程

- master_xfer---写

```
msg[0].addr = addr;          /*器件地址*/
msg[0].flags = !I2C_M_RD;  /*写标记*/
msg[0].len = count;           /*buf大小*/
msg[0].buf = &data;         /*一般有两个或多个字节组成
                第一个是目标单元，后面是写的数据*/
```

注：利用i2c_transfer发送数据只需填充一个i2c_msg即可

# 操作流程

- master_xfer---读

```
msg[0].addr = chip_addr;      /*器件地址*/
msg[0].flags = !I2C_M_RD;     /*写标记*/
msg[0].len = count;           /*buf大小*/
msg[0].buf = &addr;           /*器件单元地址*/

msg[1].addr = chip_addr;      /*器件地址*/
msg[1].flags = I2C_M_RD;      /*读标记*/
msg[1].len = count;           /*buf大小*/
msg[1].buf = &buf;            /*读取到的数据*/
```

注：利用i2c_transfer读取数据需填充两个i2c_msg

# 操作流程

② 

```
int i2c_master_send
        (struct i2c_client *client, const char *buf, int count);

int i2c_master_recv
                (struct i2c_client *client, char *buf, int count);
```

---

③ smbus方式

```
s32 i2c_smbus_read_i2c_block_data
    (struct i2c_client *client, u8 command, u8 length, u8 *values)

s32 i2c_smbus_write_block_data
(struct i2c_client *client, u8 command,u8 length, const u8*values)
```

# 操作流程

- 练习：

  参照i2c_subsys_demo，编写自己的bma150驱动

【注意】：

保证内核没有编译bma150,make menuconfig将其勾掉

```
Device Drivers  --->
  Input device support  --->
        [*]   Miscellaneous devices  --->
              <  >   BMA150 G-sensor Driver
```

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com