MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy

讲师：高焕堂（台湾）

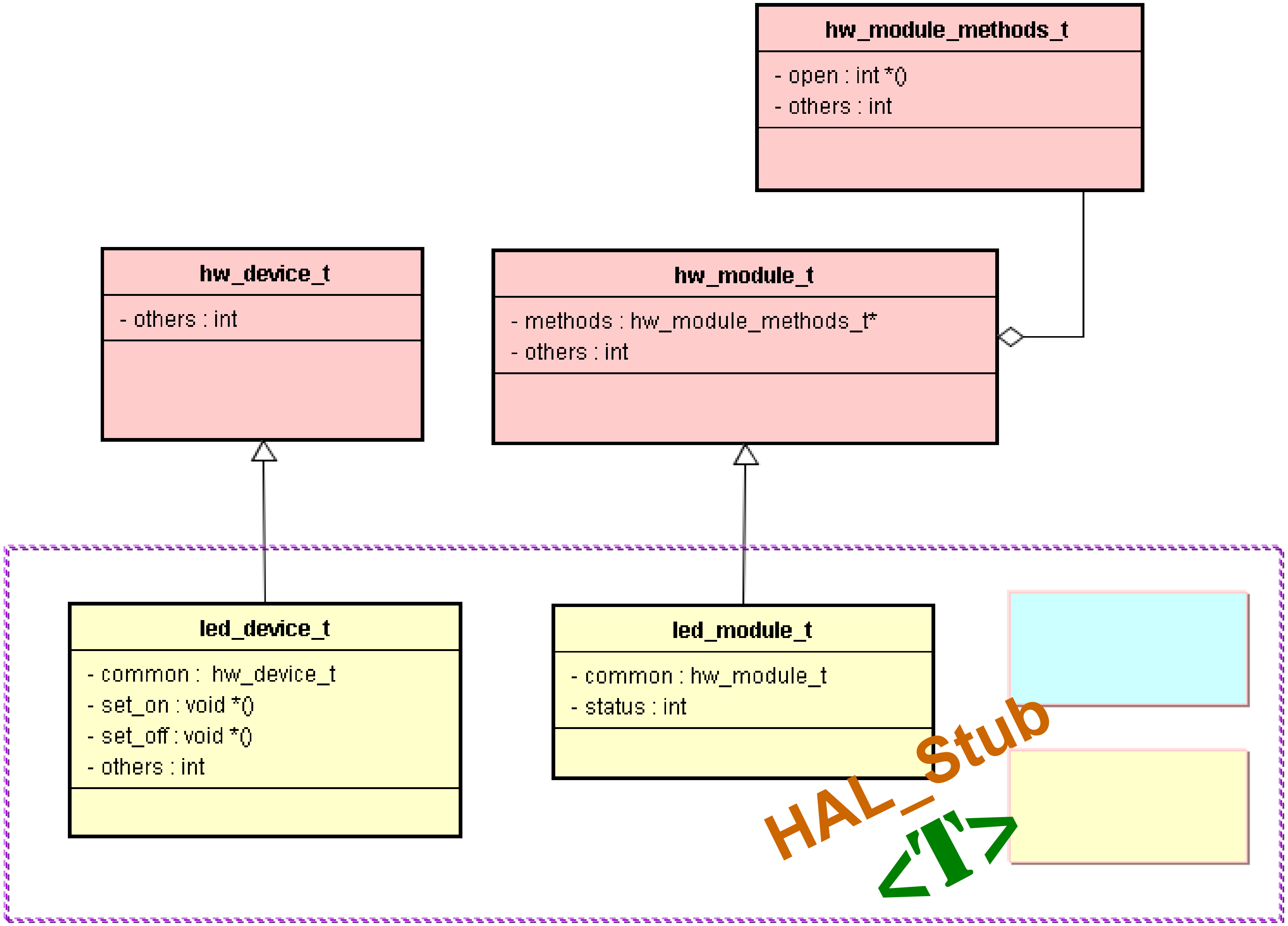
http://www.microoh.com

E02_e

HAL框架与Stub开发 (e)

By 高焕堂

扩充hw_device_t


hw_module_methods_t
- open : int *()
- others : int
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int
HAL_Stub
<'I'>

```
struct led_module_t {
   struct hw_module_t common;
   int status;
};
```

```
struct led_device_t {
   struct hw_device_t common;
   int (*set_on)(struct led_device_t *dev);
   int (*set_off)(struct led_device_t *dev);
};
```

```
static int led_device_close(struct hw_device_t* device){
        struct led_device_t* ldev =
                (struct led_device_t*)device;
        if (ldev)  free(ldev);
        return 0;
}
```

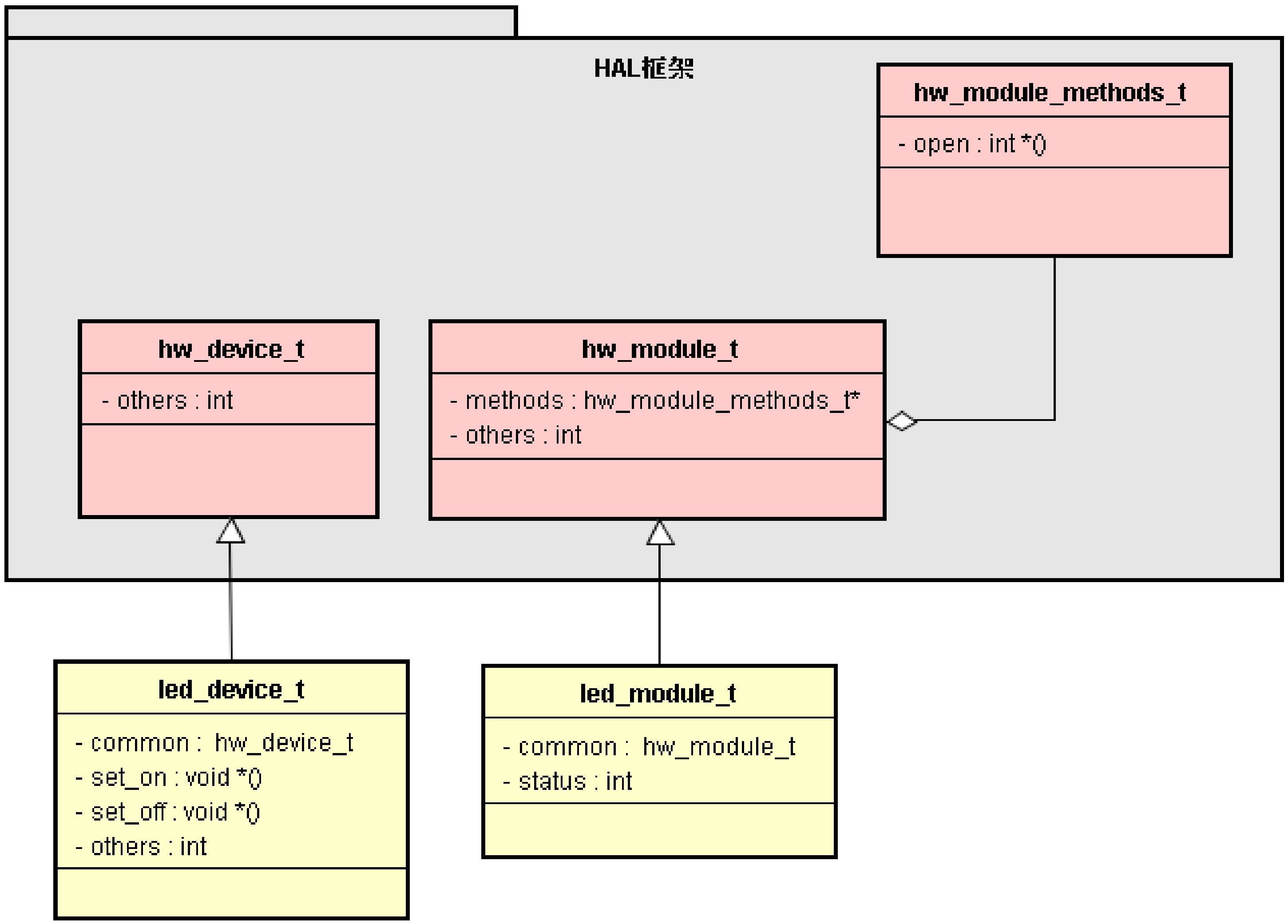
```
static int led_set_on(struct led_device_t *dev){
          // …….
          return 0;
}
static int led_set_off(struct led_device_t *dev){
          // …….
          return 0;
}
```

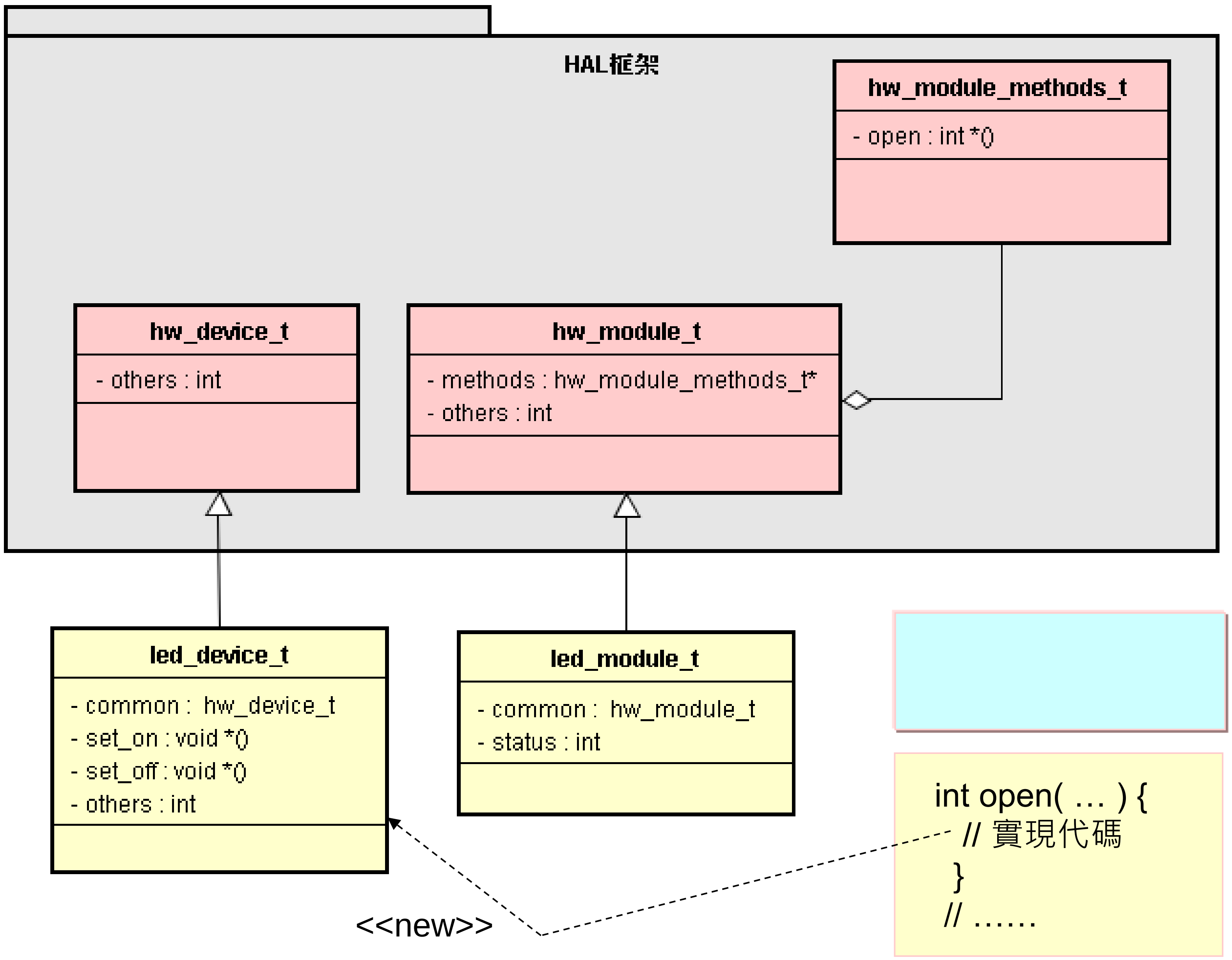
```
static int led_open(const struct hw_module_t* module, const char* name,
        struct hw_device_t** device)
{
             struct led_device_t *dev;
             LOGD("led_device_open");
             dev = (struct led_device_t*)malloc(sizeof(struct led_device_t));
             memset(dev, 0, sizeof(struct led_device_t));
             dev->common.tag =  HARDWARE_DEVICE_TAG;
             dev->common.version = 0;
             dev->common.module = (struct hw_module_t*)module;
             dev->common.close = led_device_close;             // ……
             dev->set_on= led_set_on;
             dev->device.set_off= led_set_off;
             *device = (struct hw_device_t*)dev;
             return 0;
}
```

```
static struct hw_module_methods_t  my_methods  =  {
                    open: led_open
       };
```

```
const struct led_module_t HAL_MODULE_INFO_SYM = {
    common: {
        tag: HARDWARE_MODULE_TAG,
        version_major: 1,
        version_minor: 0,
        id: LED_HARDWARE_MODULE_ID,
        name: “Test LED Stub",
        author: “Test Project Team",
        methods: &my_methods,
    }
     status: -1,
};
```

谁来创建led_device_t的对象呢?


HAL框架
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int

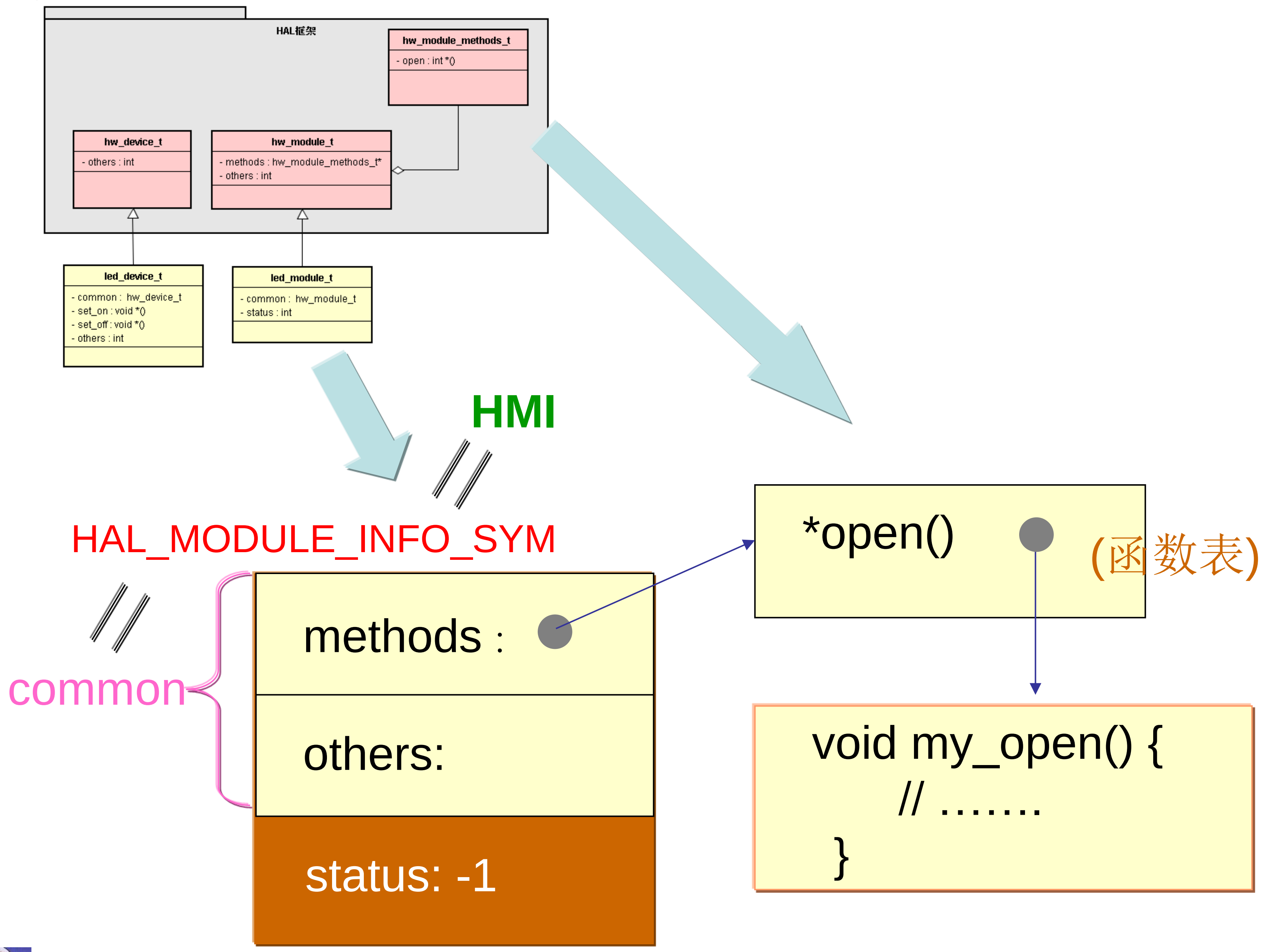

HAL框架
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int
<<new>>
int open(…) {
// 實現代碼
}
// ……

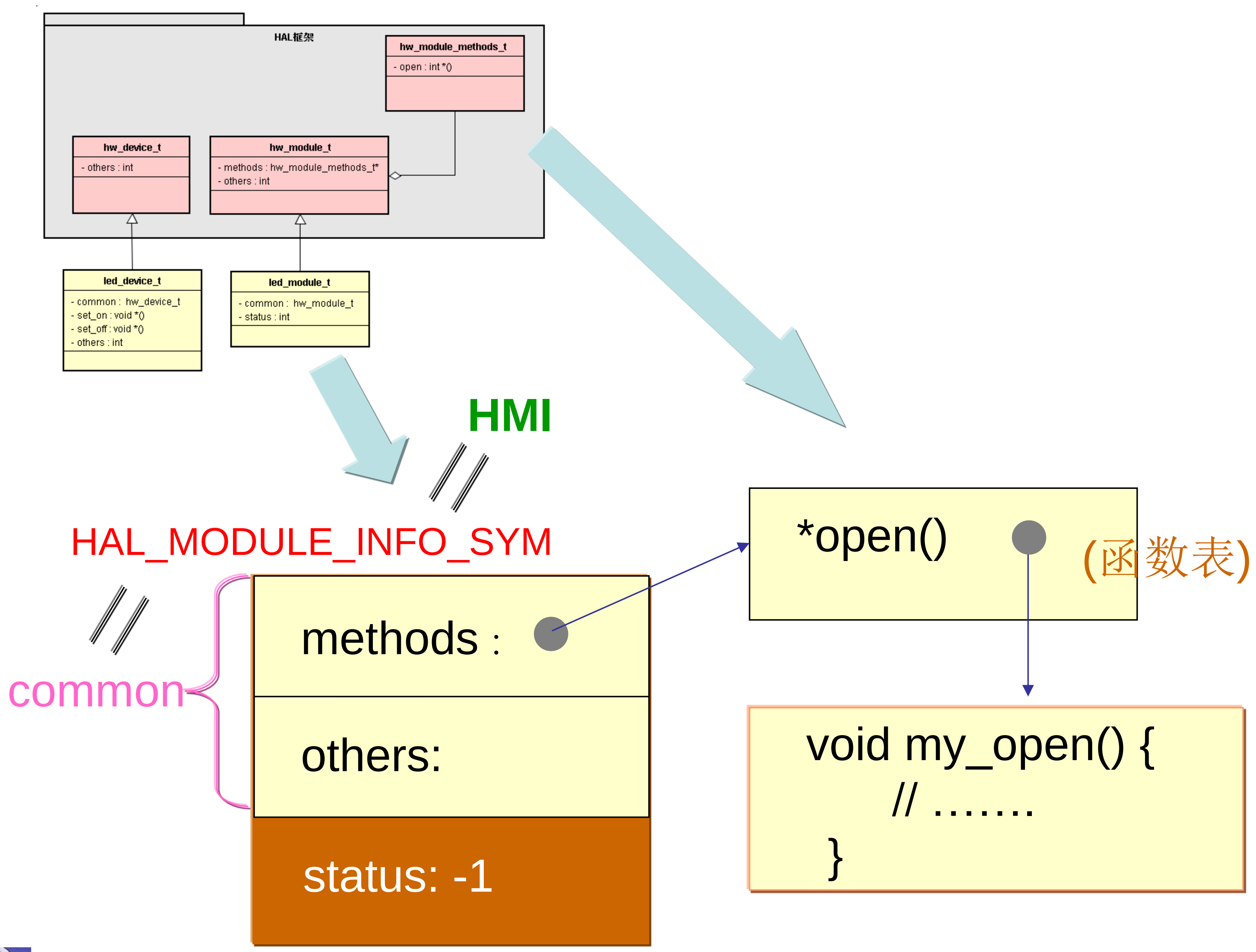
```
static int led_open(const struct hw_module_t* module, const char* name,
        struct hw_device_t** device)
{
            struct led_device_t *dev;
            dev = (struct led_device_t*)malloc(sizeof(struct led_device_t));
            memset(dev, 0, sizeof(struct led_device_t));
            dev->common.tag =  HARDWARE_DEVICE_TAG;
            dev->common.version = 0;
            dev->common.module = (struct hw_module_t*)module;
            dev->common.close = led_device_close;            // ……
            dev->set_on= led_set_on;
            dev->device.set_off= led_set_off;
            *device = (struct hw_device_t*)dev;
            return 0;
}
```

- 写好了上述的HAL-Stub代码，就能编译&连结成为*.so文檔。

- 载入*.so文檔，执行这些HAL-Stub代码，在run-time就创建对象，并设定函数指针，如下图：

Run-time


HAL框架
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int
HMI
HAL_MODULE_INFO_SYM
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}

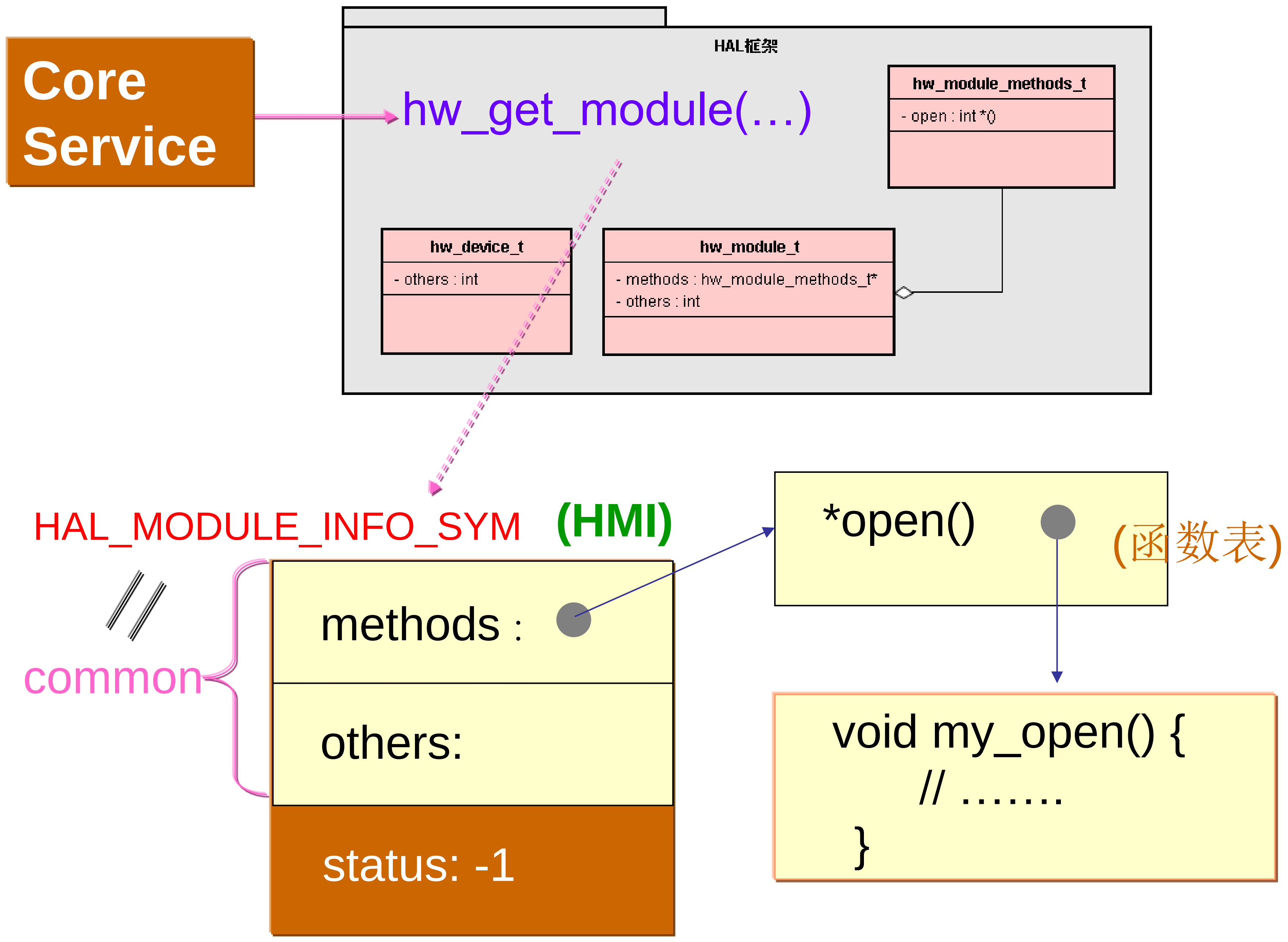

HAL框架
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int
HMI
//
HAL_MODULE_INFO_SYM
//
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}

Client使用HAL的第1个步骤

- HAL框架提供了一个公用的函数：
- hw_get_module(const char *id, const struct hw_module_t **module)

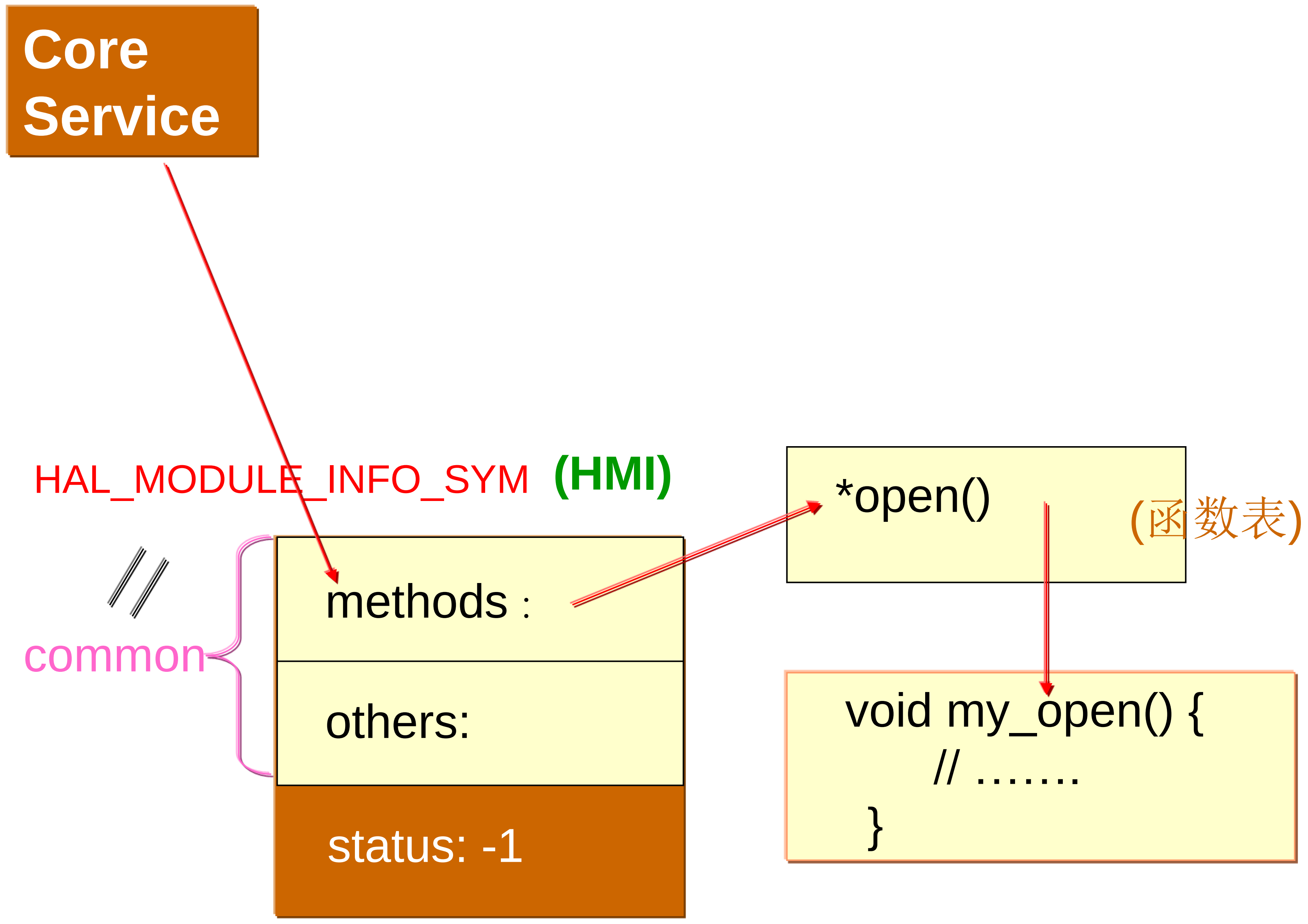
- 这个函数的主要功能是根据模块ID(module_id)去查找注册在当前系统中与id对应的硬件对象，然后载入(load)其相应的HAL层驱动模块的*so文件。

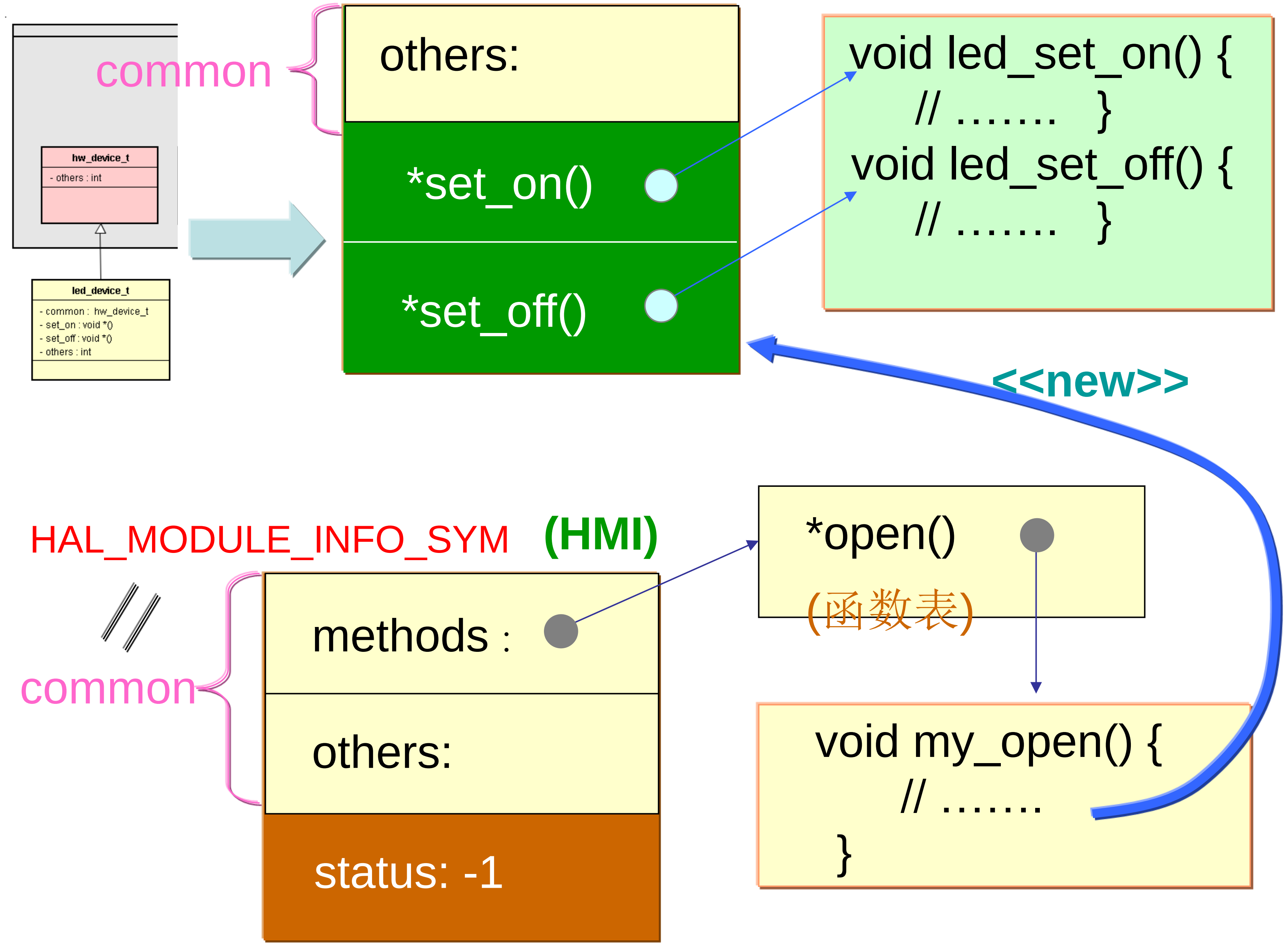
- 從*.so里查找”HMI”这个符号，如果在so代码里有定义的函数名或变量名为HMI，返回其地址。

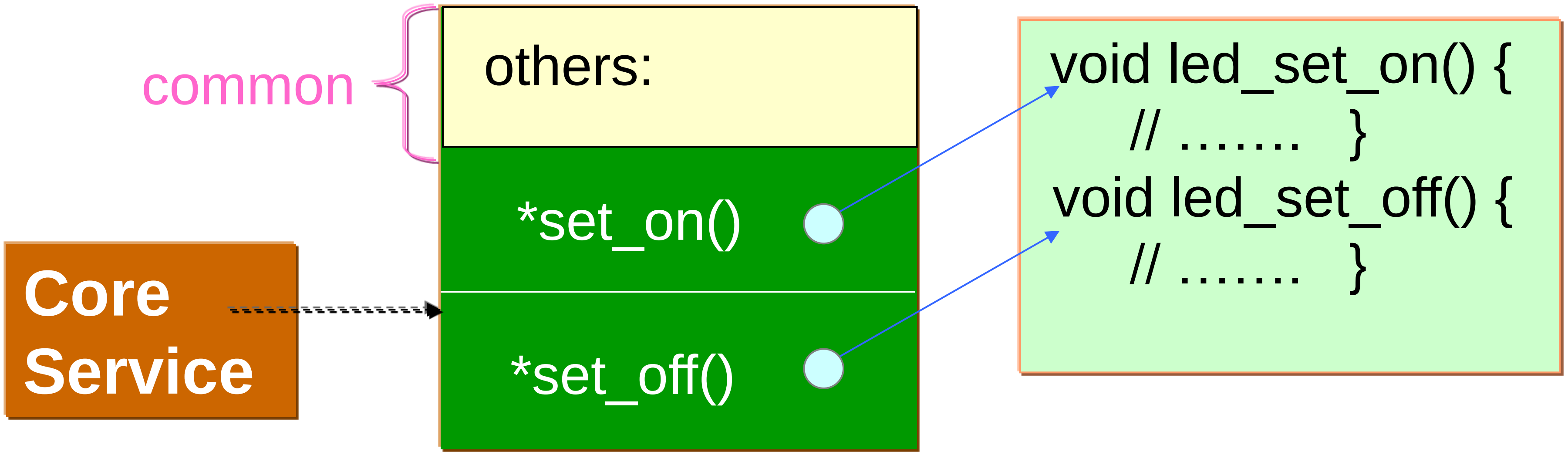

Core Service
HAL框架
hw_get_module(…)
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
HAL_MODULE_INFO_SYM
(HMI)
//
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}

- 從*.so里查找”HMI”这个符号，如果在so代码里有定义的函数名或变量名为HMI，返回其地址。

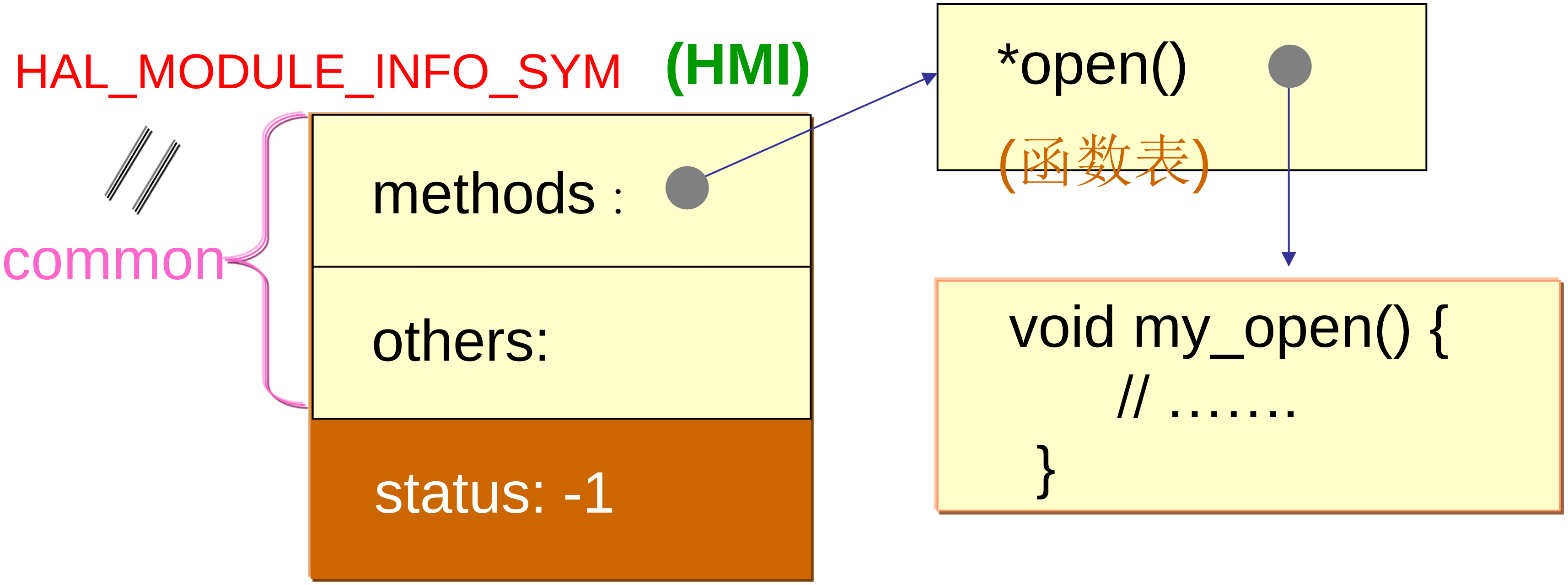
Client使用HAL的第2个步骤


Core Service
HAL_MODULE_INFO_SYM
(HMI)
//
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}


common
hw_device_t
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
others:
*set_on()
*set_off()
void led_set_on() {
// ……. }
void led_set_off() {
// ……. }
<<new>>
HAL_MODULE_INFO_SYM
(HMI)
//
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}

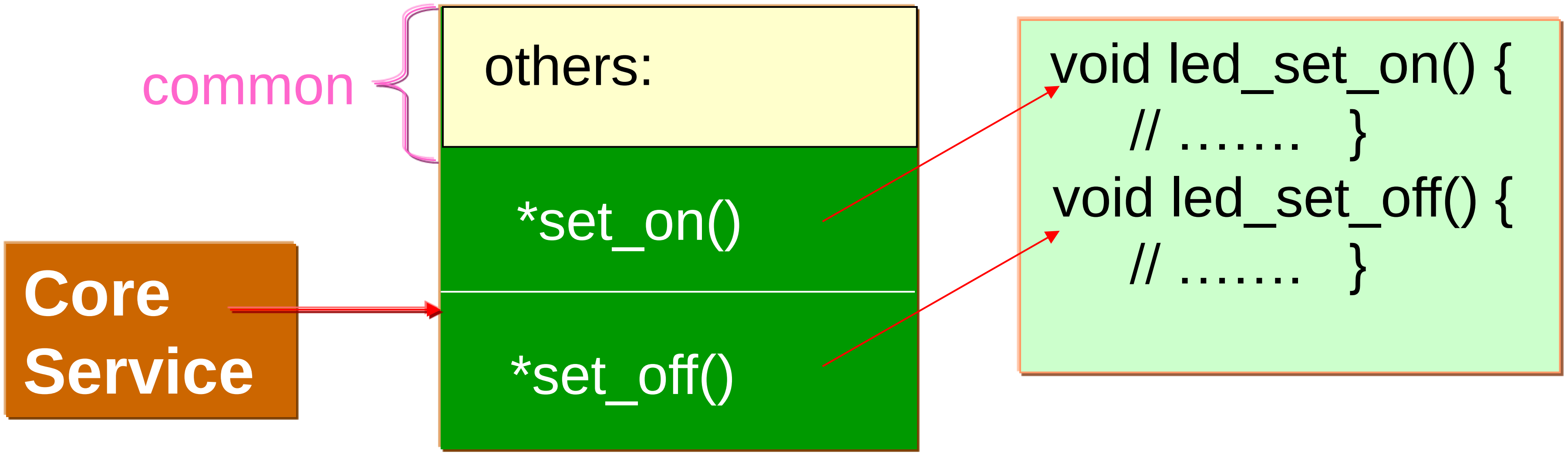


```
void led_set_on() {
     // …….   }
void led_set_off() {
     // …….   }
```

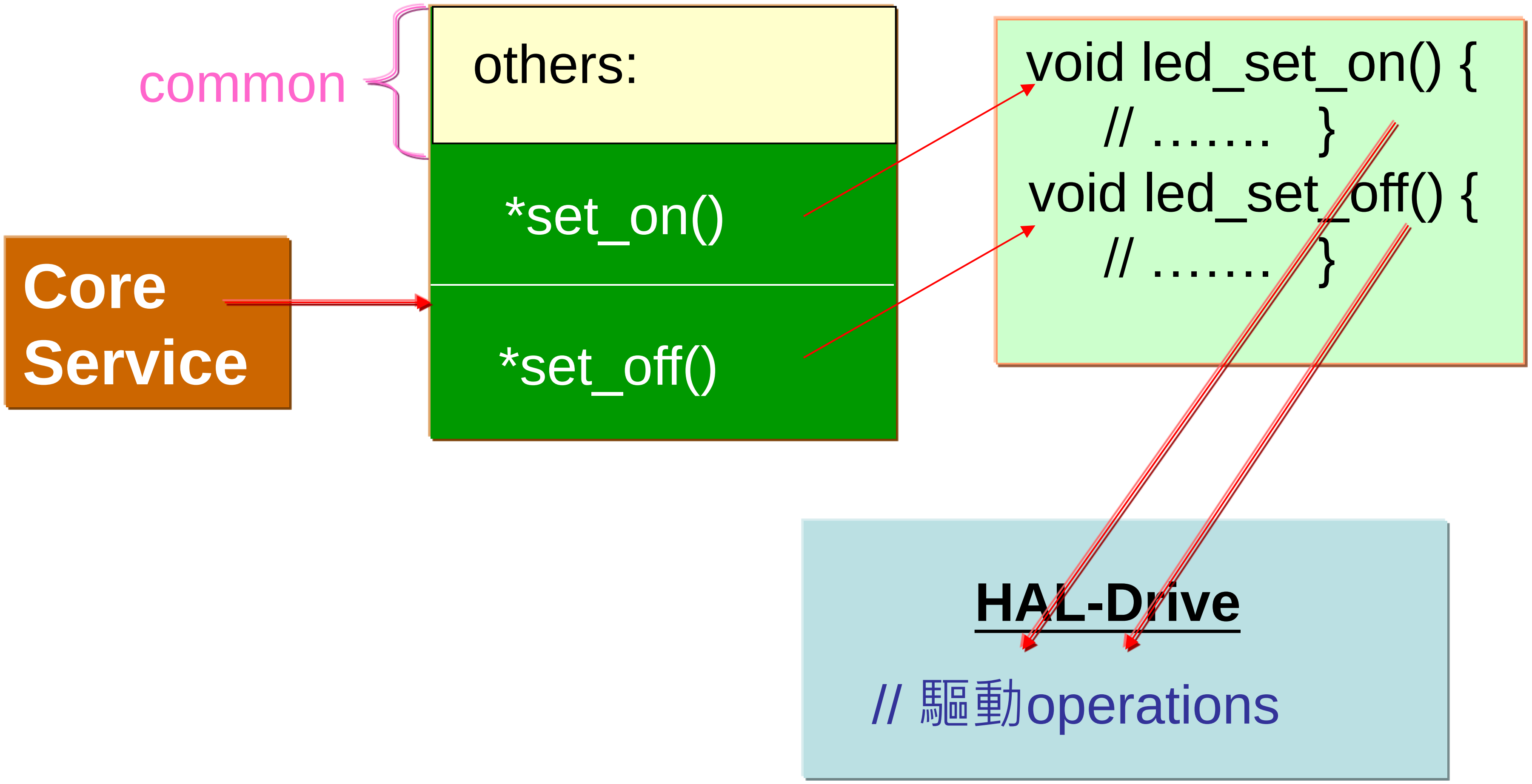


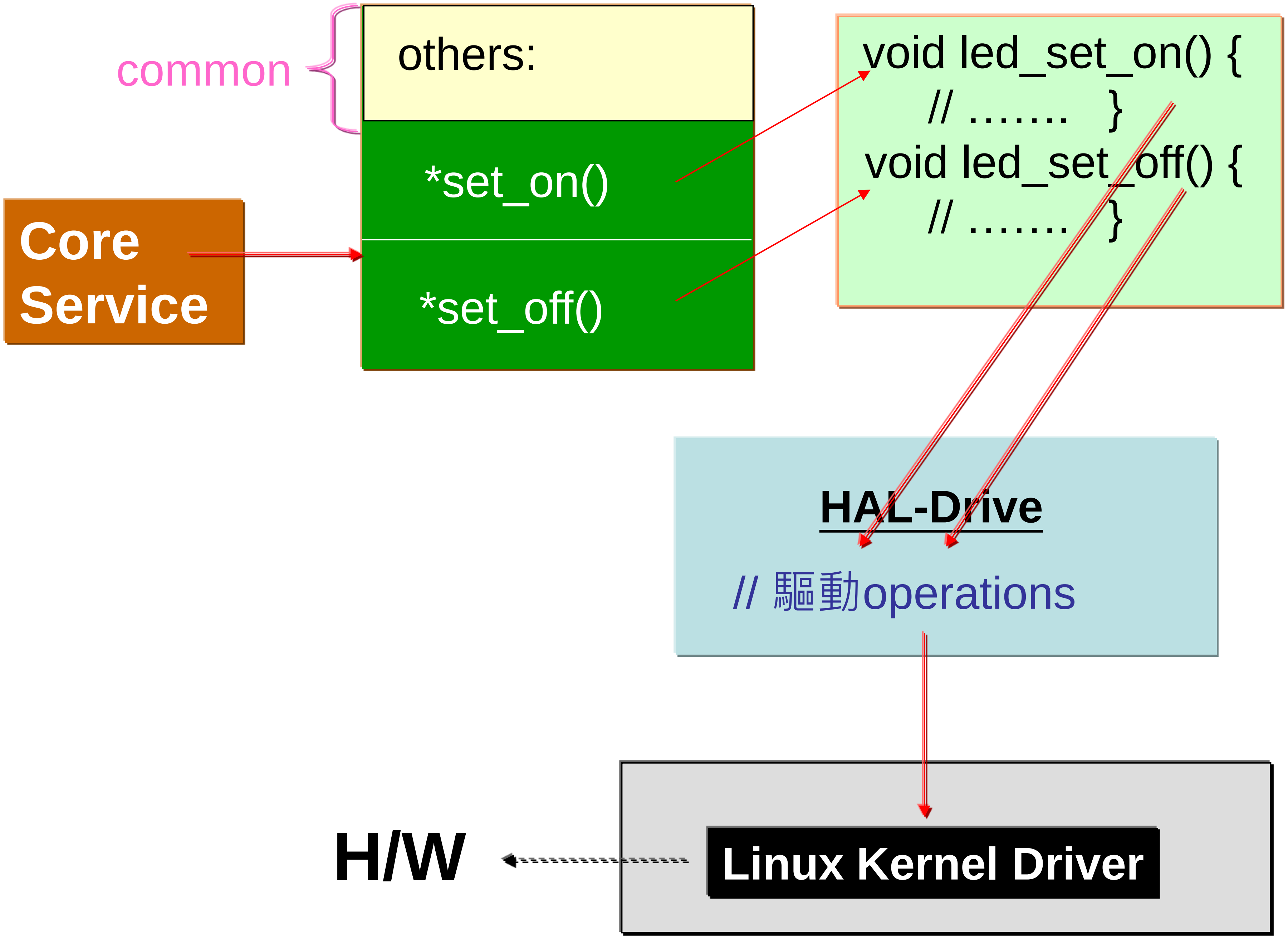
```
void my_open() {
     // …….
  }
```

Client使用HAL的第3个步骤



```
void led_set_on() {
    // …….   }
void led_set_off() {
    // …….   }
```


common
others:
*set_on()
*set_off()
Core Service
void led_set_on() {
// ……. }
void led_set_off() {
// ……. }
HAL-Drive
// 驅動operations


common
others:
*set_on()
*set_off()
Core Service
void led_set_on() {
// ……. }
void led_set_off() {
// ……. }
HAL-Drive
// 驅動operations
H/W
Linux Kernel Driver

~ Continued ~