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）

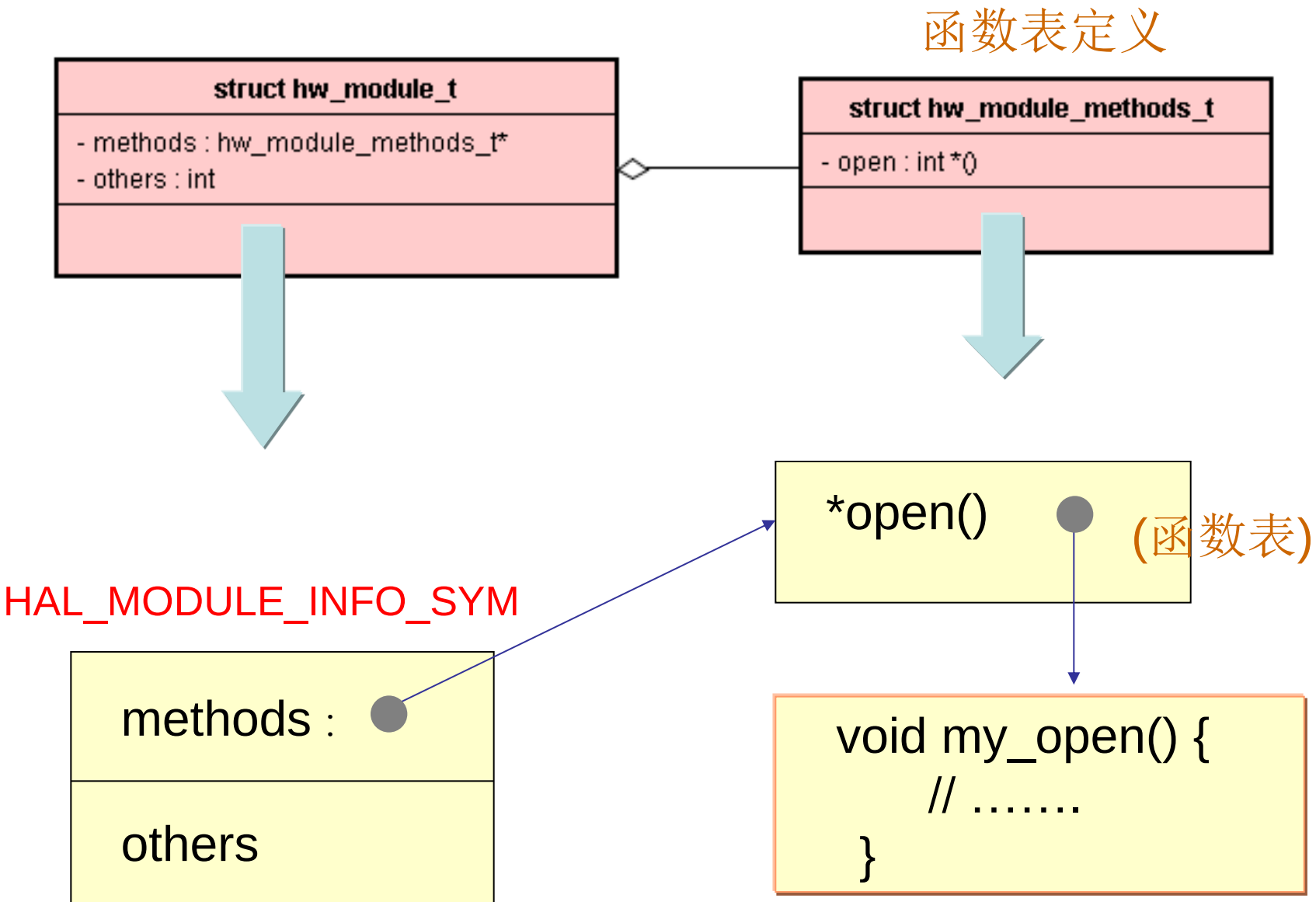
http://www.microoh.com

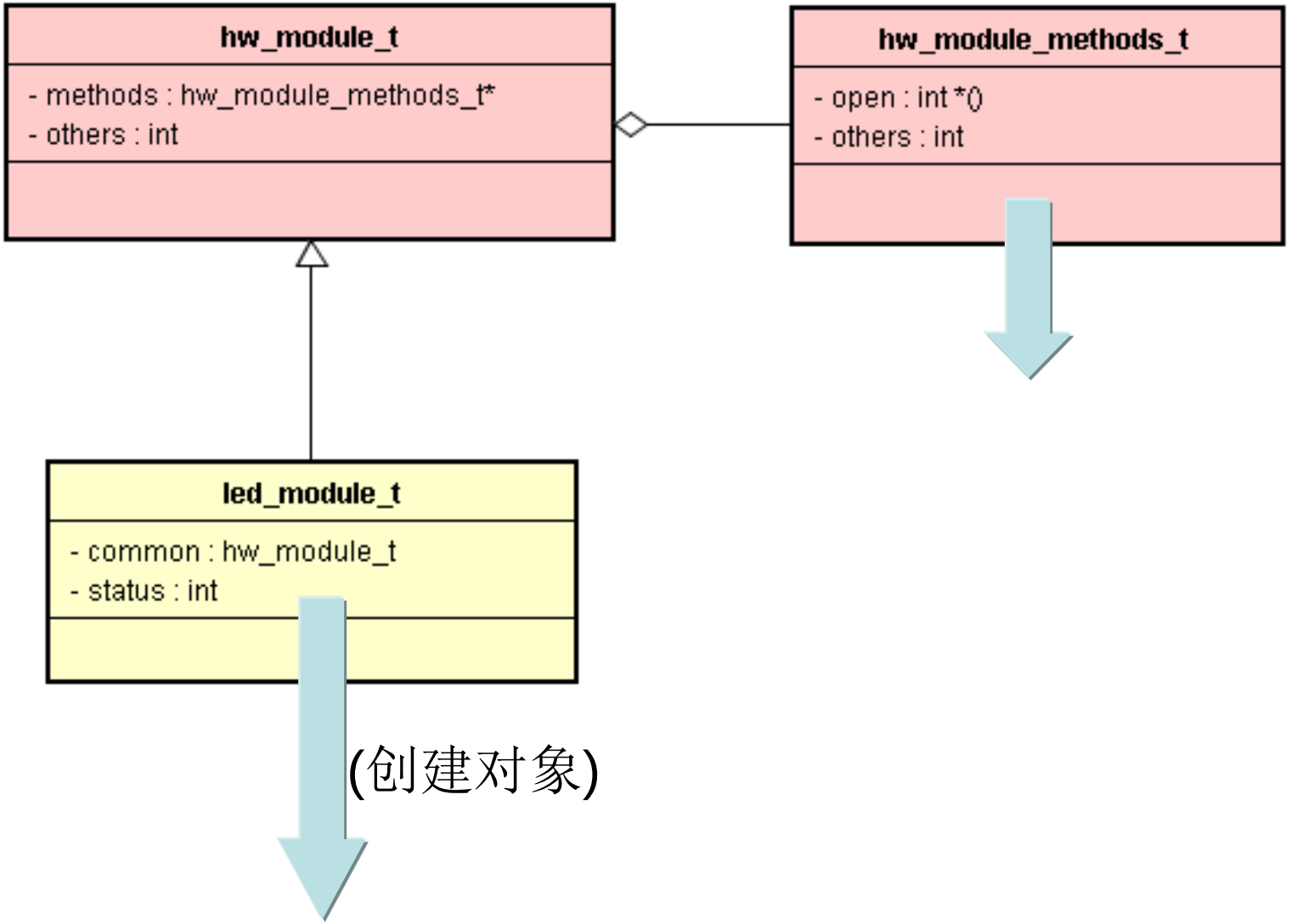
E02_d

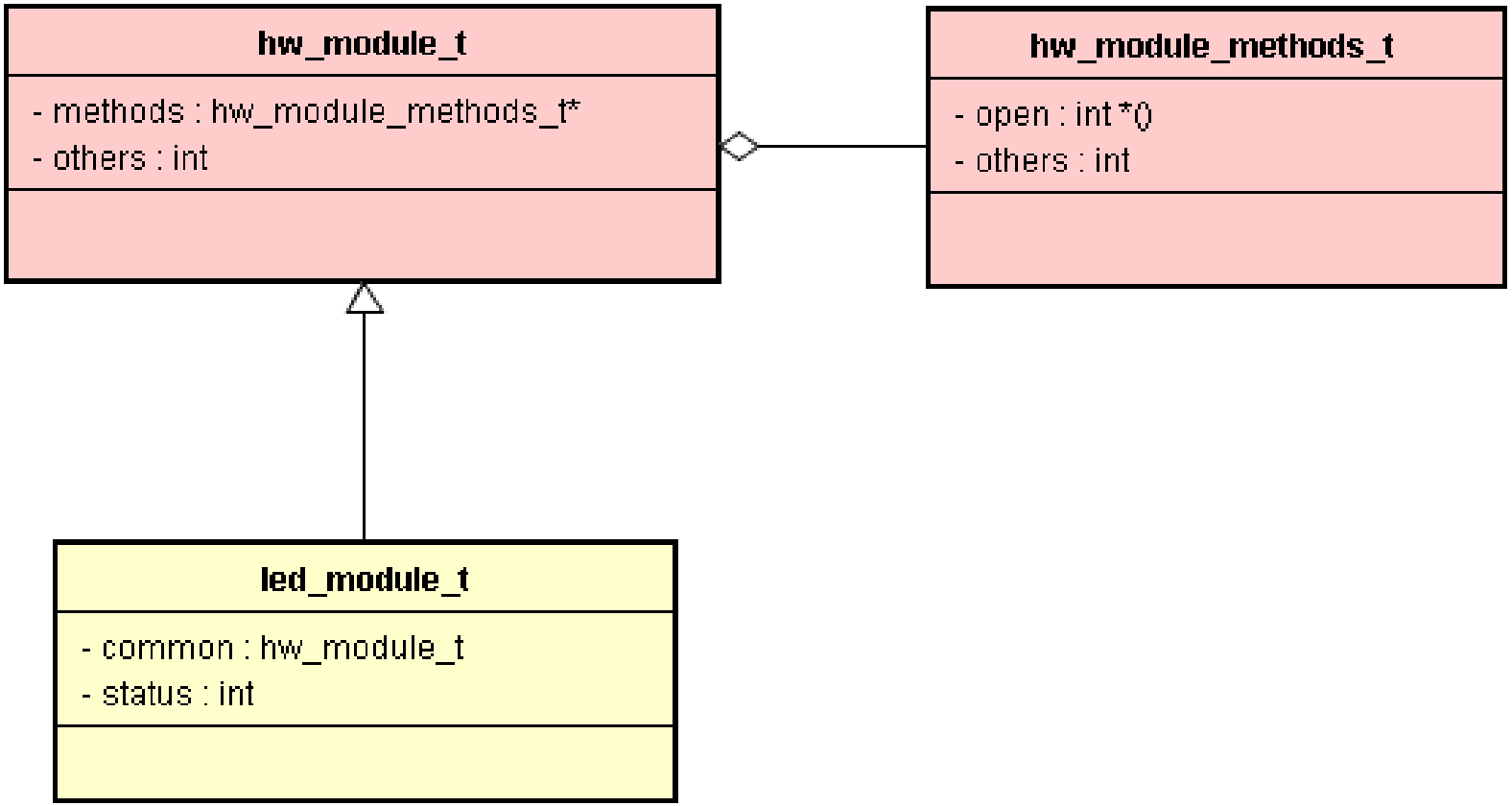
HAL框架与Stub开发 (d)

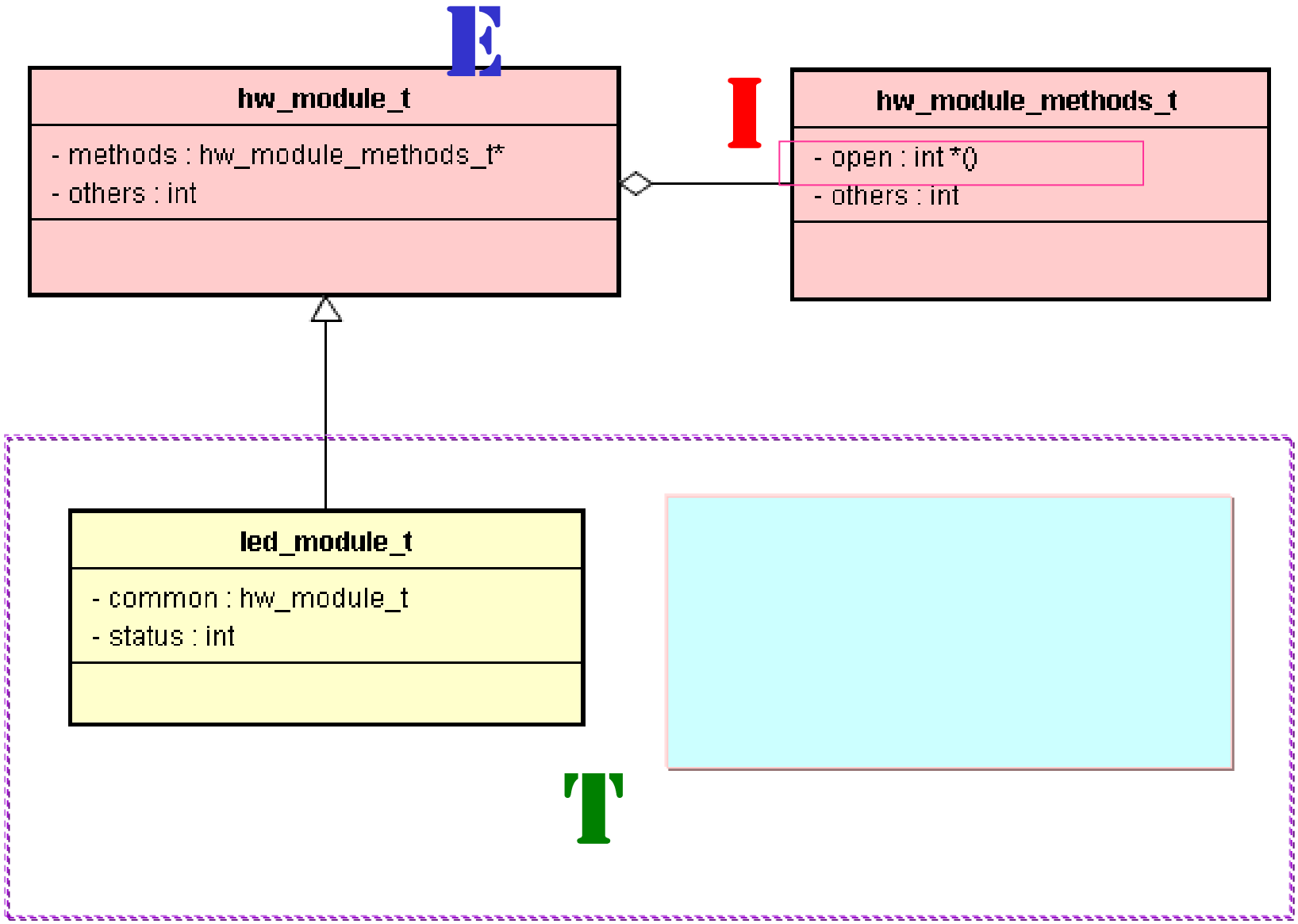
By 高煥堂

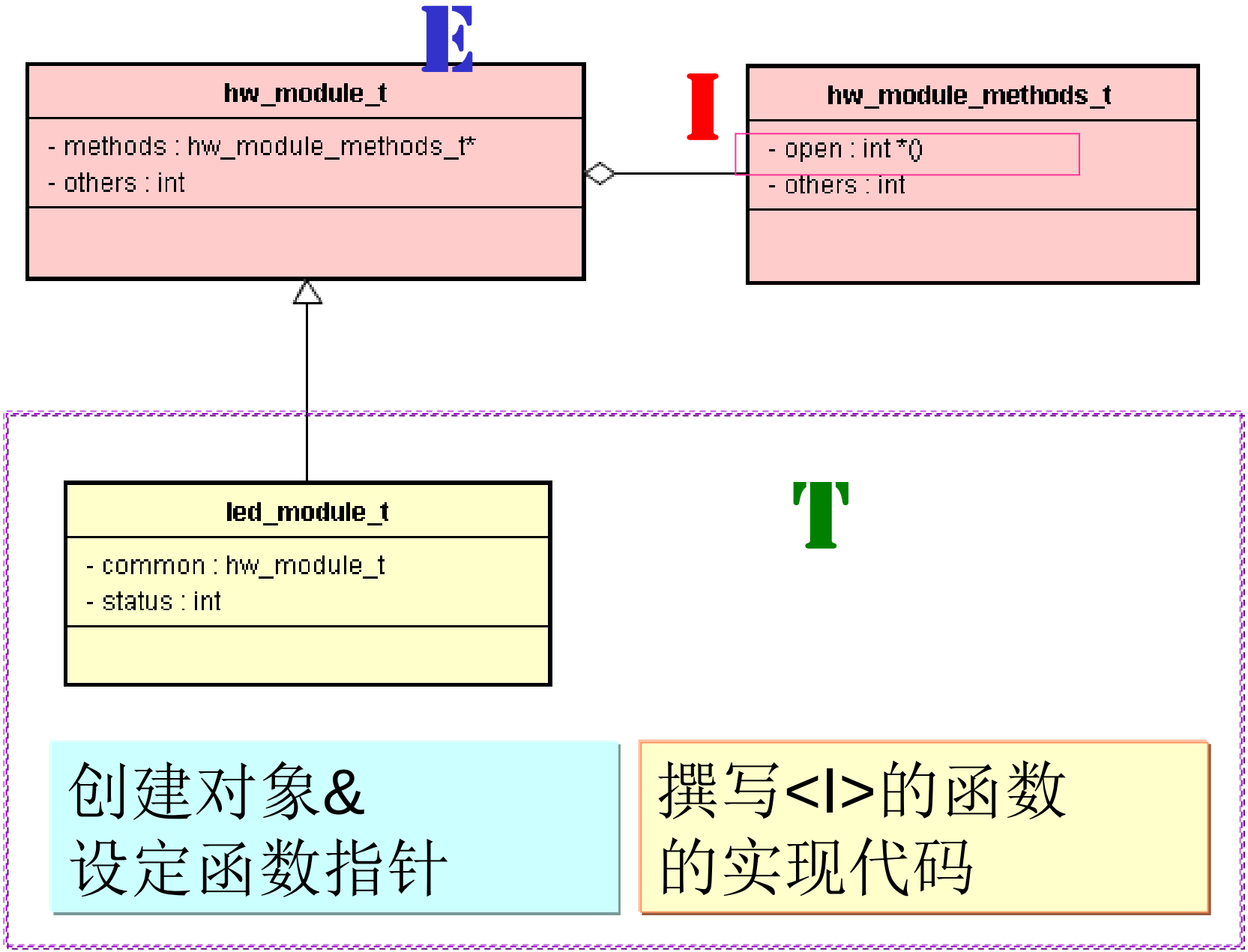
前言


函数表定义
struct hw_module_t
- methods : hw_module_methods_t*
- others : int
struct hw_module_methods_t
- open : int *()
*open()
(函数表)
HAL_MODULE_INFO_SYM
methods :
others
void my_open() {
// .....
}

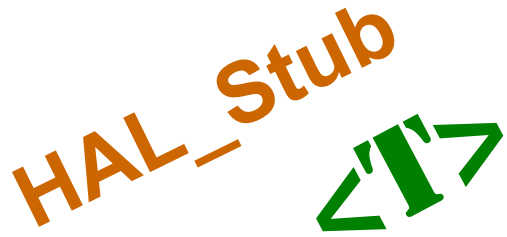

hw_module_t
- methods : hw_module_methods_t*
- others : int
hw_module_methods_t
- open : int *()
- others : int
led_module_t
- common : hw_module_t
- status : int
(创建对象)


hw_module_t
- methods : hw_module_methods_t*
- others : int
hw_module_methods_t
- open : int *()
- others : int
led_module_t
- common : hw_module_t
- status : int


E
hw_module_t
- methods : hw_module_methods_t*
- others : int
I
hw_module_methods_t
- open : int *()
- others : int
led_module_t
- common : hw_module_t
- status : int
T


E
hw_module_t
- methods : hw_module_methods_t*
- others : int
I
hw_module_methods_t
- open : int *()
- others : int
led_module_t
- common : hw_module_t
- status : int
T
创建对象&
设定函数指针
撰写<I>的函数
的实现代码

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




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

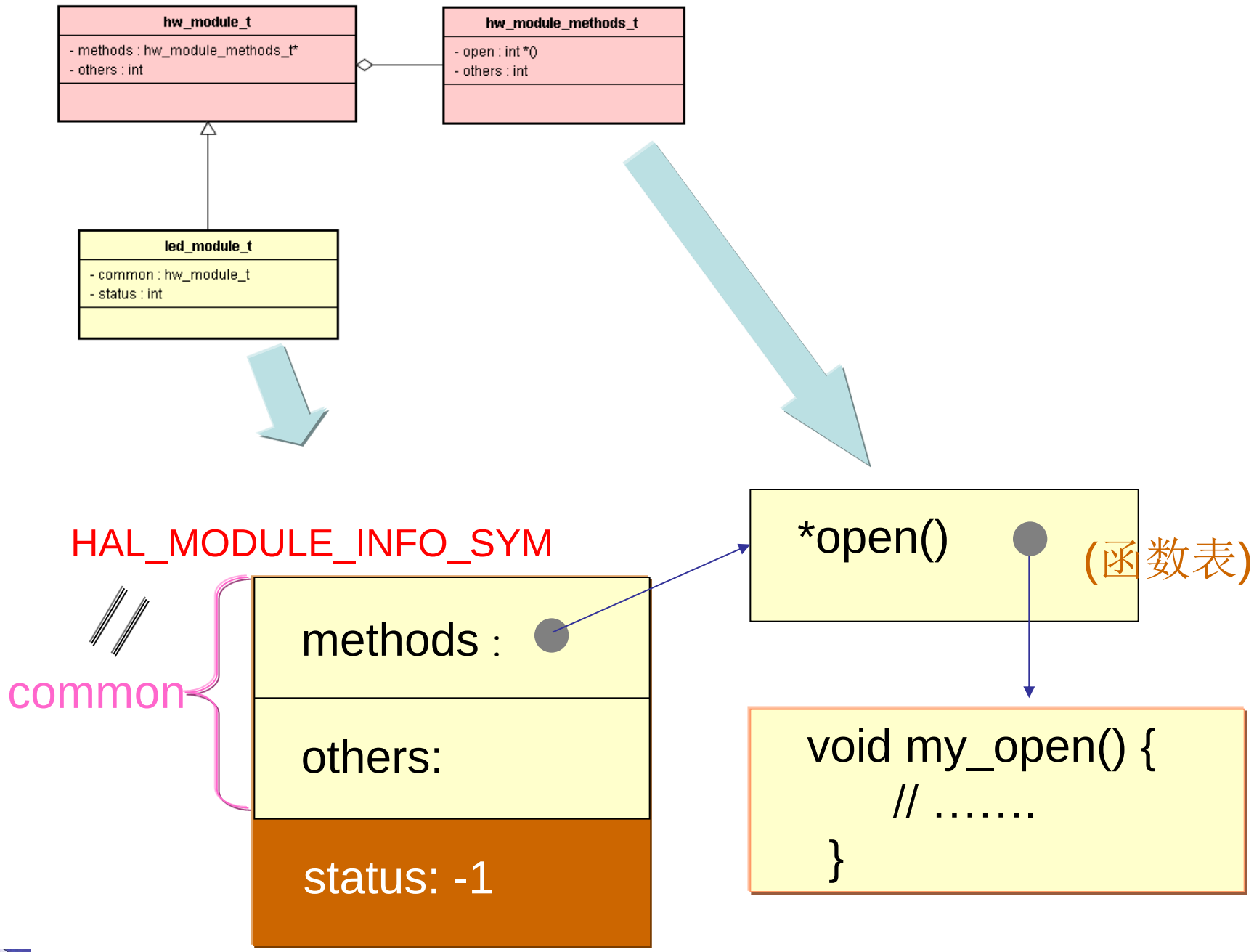
```
const struct led_module_t
HAL_MODULE_INFO_SYM = {
    common: {
        // ……..
        methods: &my_methods,
     }
     status: -1,
};
```

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

- 写好了上述的HAL-Stub代码，就能编译&连结成为*.so文檔。

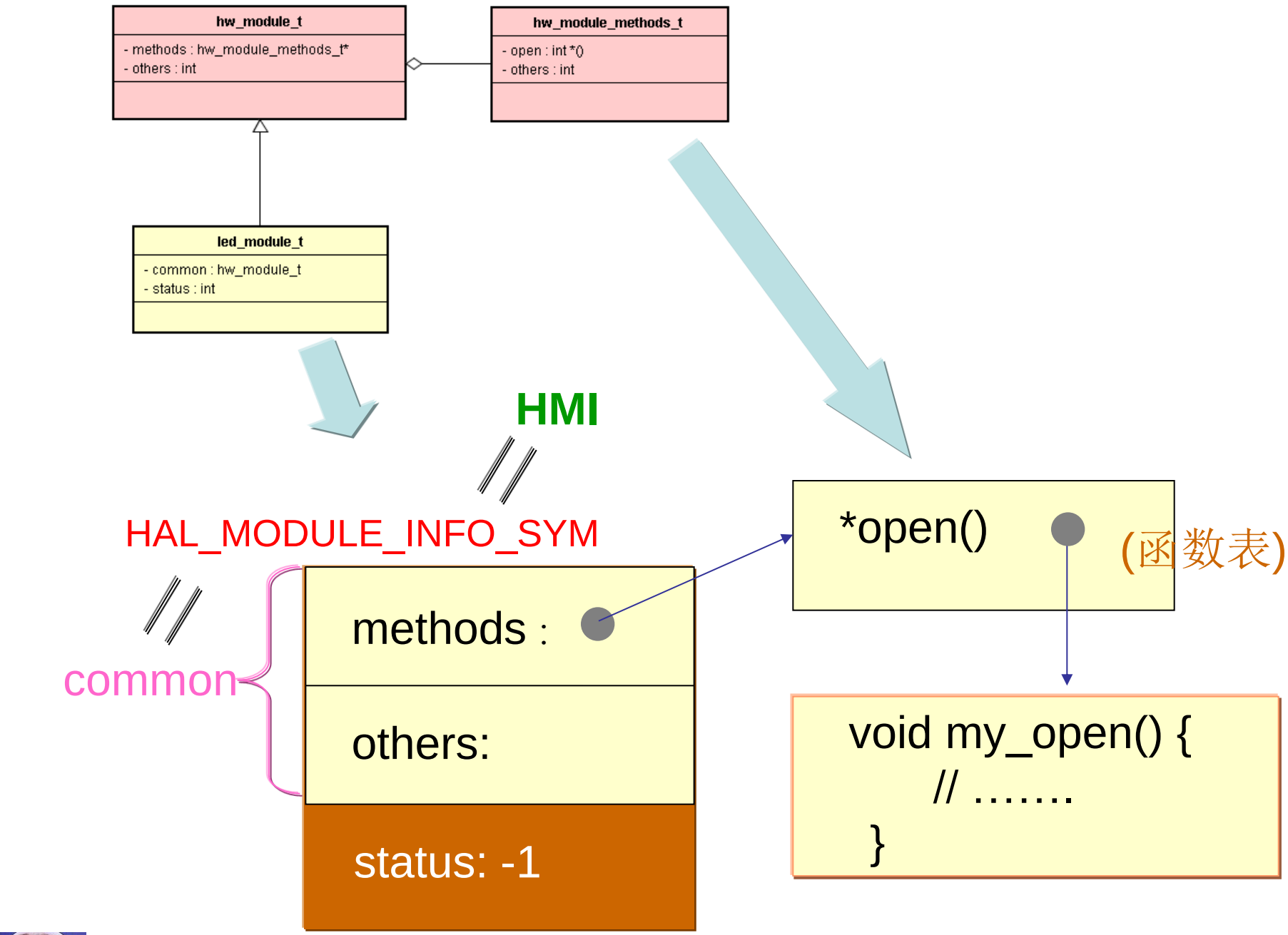
- 载入*.so文檔，执行这些HAL-Stub代码，在run-time就创建对象，并设定函数指针，如下图：

Run-time

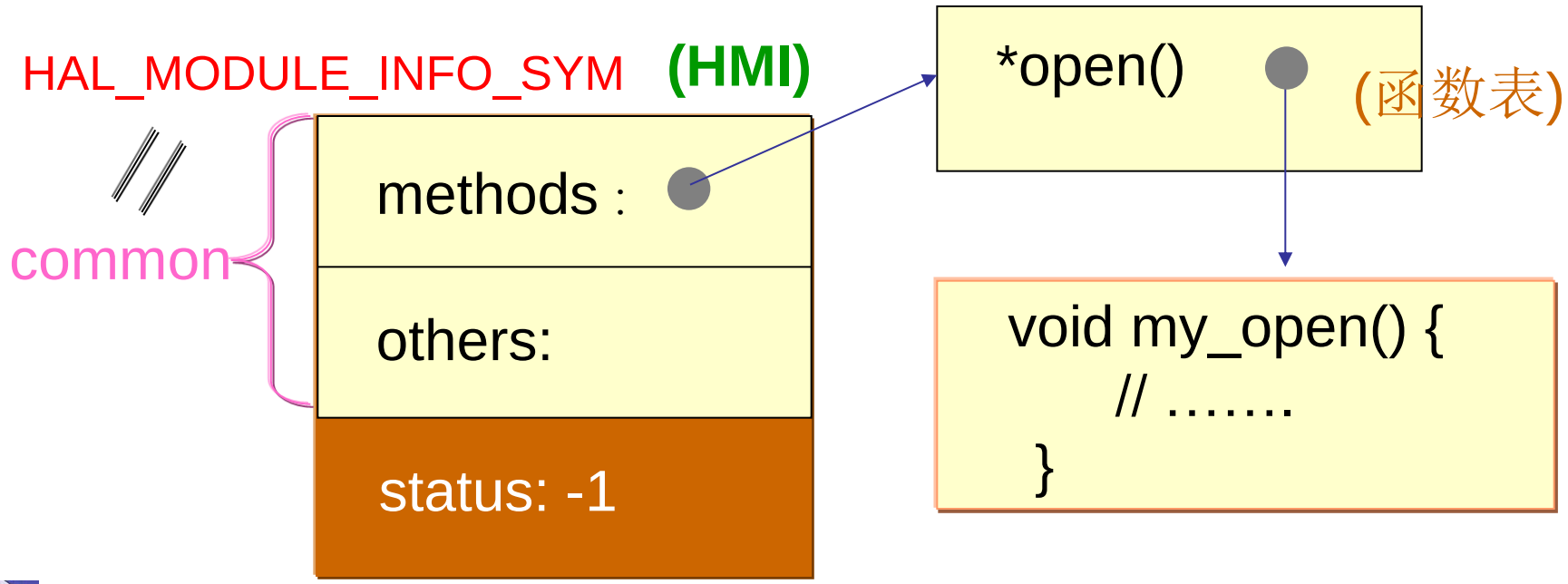

hw_module_t
- methods : hw_module_methods_t*
- others : int
hw_module_methods_t
- open : int *()
- others : int
led_module_t
- common : hw_module_t
- status : int
HAL_MODULE_INFO_SYM
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// .....
}

- 在HAL框架里，定义了如下：

```
#define HAL_MODULE_INFO_SYM  HMI
#define HAL_MODULE_INFO_SYM_AS_STR  "HMI"
```


hw_module_t
- methods : hw_module_methods_t*
- others : int
hw_module_methods_t
- open : int *()
- 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() {
// .....
}

```
const struct led_module_t
HAL_MODULE_INFO_SYM = {
    common: {
        // ……..
        methods: &my_methods,
     }
     status: -1,
};
```

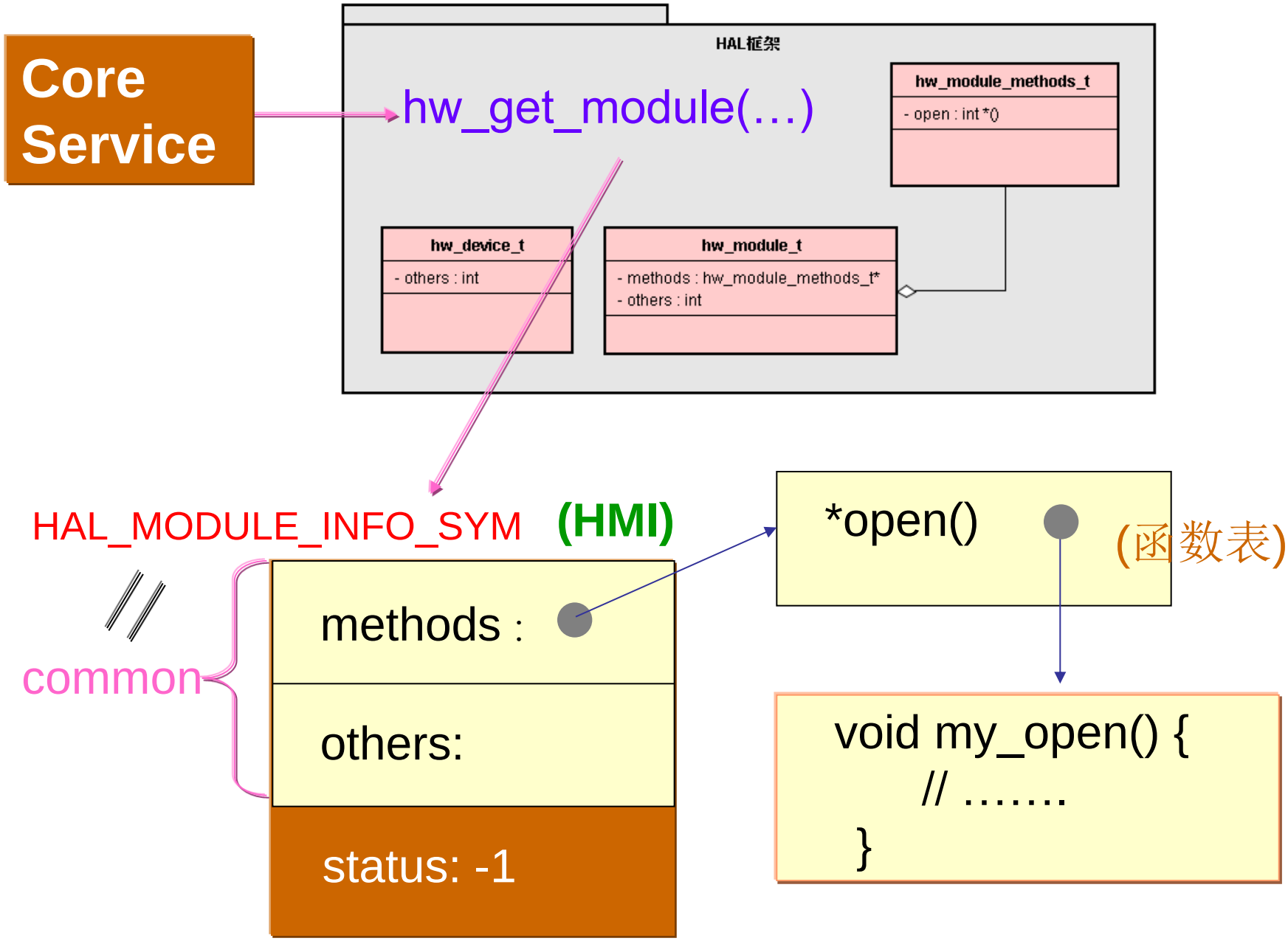


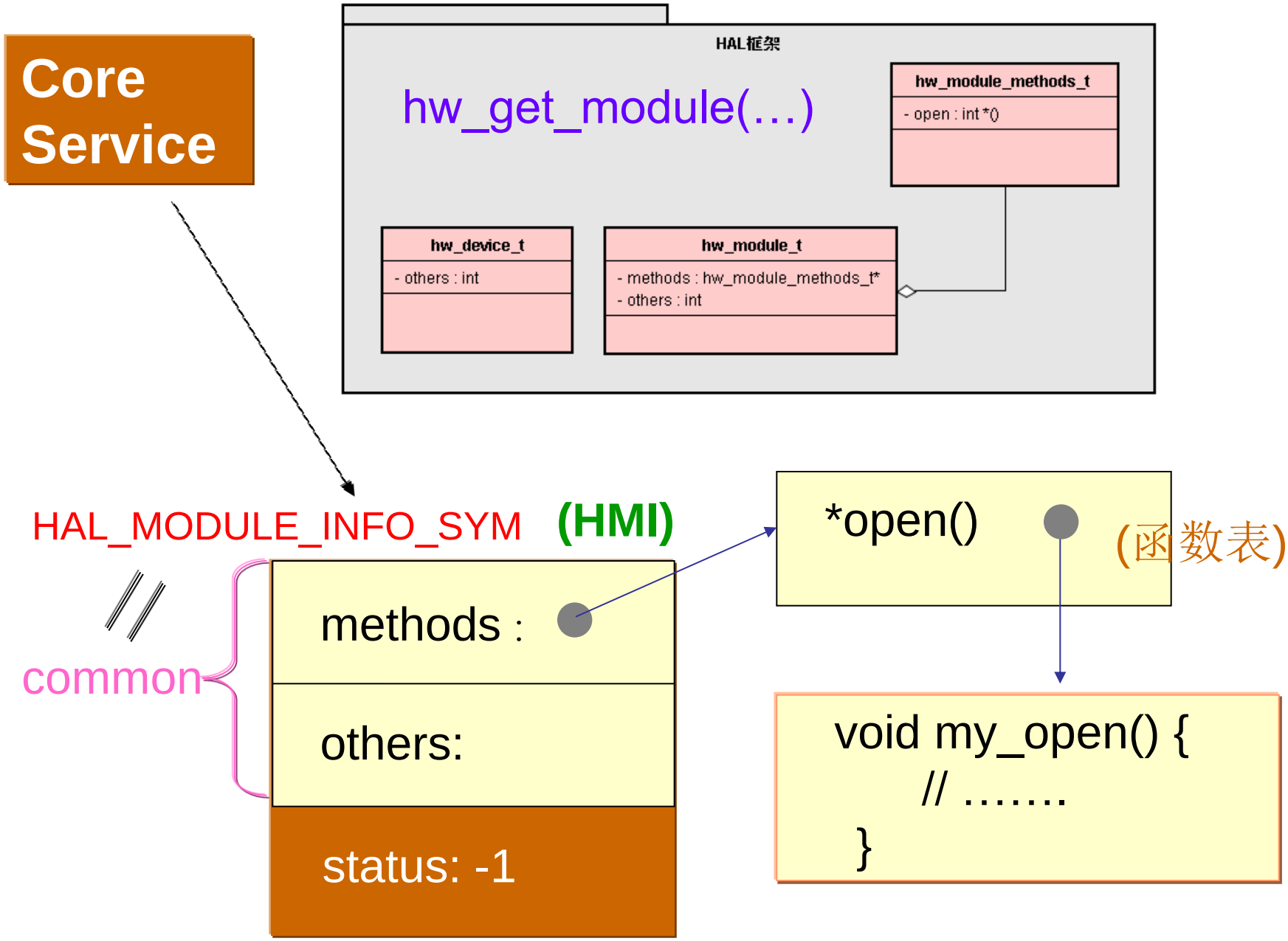
Client使用HAL的第1个步骤

- 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。

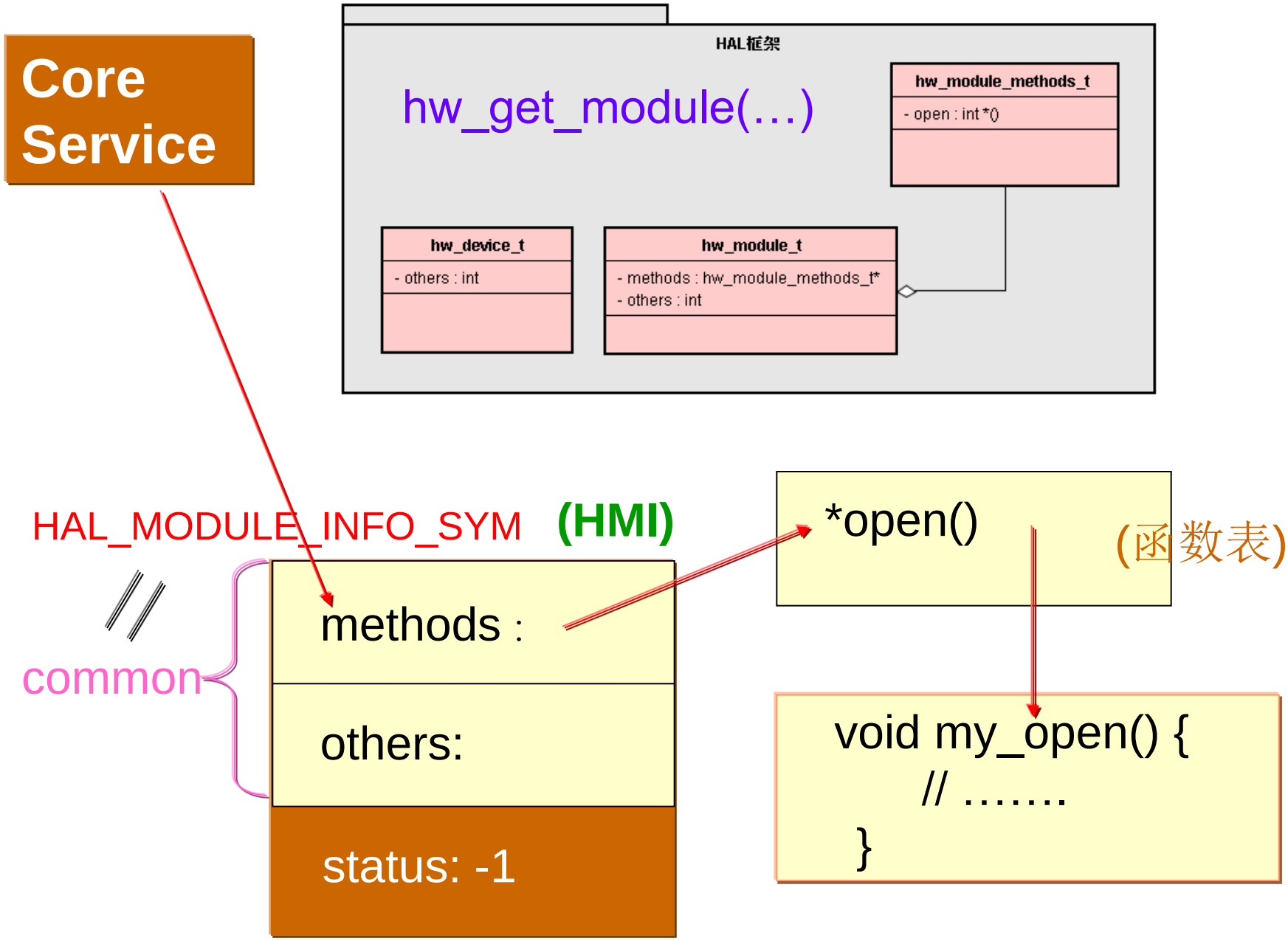
- 從*.so里查找”HMI”这个符号，如果在so代码里有定义的函数名或变量名为HMI，返回其地址。


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() {
// …….
}


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() {
// …….
}

~ Continued ~