BabyOS 如何新增驱动

1. 设备与驱动

BabyOS 主要由两大部分组成，功能模块和设备驱动。本文将先介绍设备驱动部分的设计思路，帮助开发者更好的添加驱动代码。如果发现设计中存在问题也希望及时指出。

BabyOS 设备的结构如下图所示

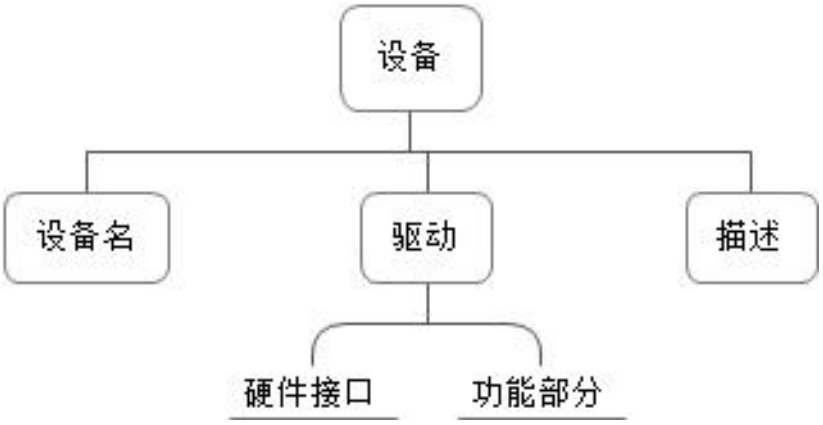


图 1-1 设备与驱动

BabyOS 所使用的设备都是需要在 b_device_list.h 中进行注册，注册时指定设备名、驱动及设备的描述。例如使用 SPI FLASH，需要按照如下方式进行注册：

```
B_DEVICE_REG(W25QXX, bW25X_Driver, "flash")
```

设备名 W25QXX 这个如何使用呢，例如 b_kv 需要 SPI FLASH，那么在初始化 b_kv 时需要指定其使用的设备，那么就传入这个设备名：

```
bKV_Init(W25QXX, 0xA000, 4096 * 4, 4096)
```

驱动又分为硬件接口和功能两个部分。以 SPI FLASH 为例，硬件接口部分指 SPI 和 CS 引脚控制，功能部分指 SPI FLASH 基于硬件接口之上实现的相关操作（休眠、唤醒、读、写、擦除等）。

之所以这样分开是因为在实际项目中会涉及到一个 MCU 连接多个相同设备的场景，例如使用了 2 个 SPIFLASH，那么这两个 SPIFLASH 设备的驱动可以共用功能部分代码，只需要各自实现硬件接口部分代码即可。

BabyOS 的设备驱动有统一的接口：

```
typedef struct bDriverInterface
{
    int (*init)(struct bDriverInterface *);
    int (*open)(void);
    int (*close)(void);
    int (*ctl)(uint8_t cmd, void *param);
    int (*write)(uint32_t addr, uint8_t *pbuf, uint16_t len);
    int (*read)(uint32_t addr, uint8_t *pbuf, uint16_t len);
    void *_private;
}bDriverInterface_t;
```

上述结构中，_private 是给硬件接口和私有变量准备。

设备驱动的 h 文件中会有一个以 Private_t 结尾的结构体，这个结构体便是用来放硬件接口和私有变量的。

```
typedef struct
{
    uint8_t (*pSPI_ReadWriteByte)(uint8_t);
    void (*pCS_Control)(uint8_t);
}bW25X_Private_t;
```

接下来就是如何将驱动的功能部分和硬件接口部分结合成为一个完整的驱动。在 b_driver.h 有如下所示的宏便是来做这件事的：

```
#define NEW_W25X_DRV(name, hal)
```

所以一个完整的 SPI FLASH 驱动是这样形成的：

```
NEW_W25X_DRV(bW25X_Driver, bW25X_Private); //形成驱动 bW25X_Driver
```

设备的第三个参数是描述设备的字符串

2. 新增驱动

2.1. 创建临时分支

在 dev 分支基础上新建分支用于调试，在 dev 分支下执行如下指令创建新分支 fm25cl。

git checkout -b fm25cl

2.2. 新建 c 文件与 h 文件

新建 C 文件和 h 文件分别放在 bos/drivers/src 和 bos/drivers/inc 目录下。添加.c 至工程。

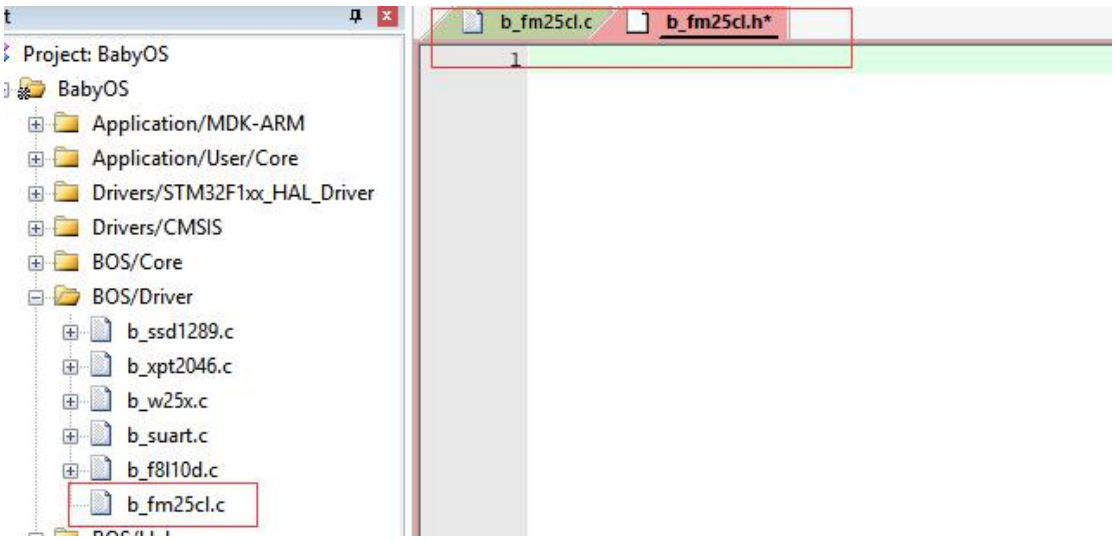


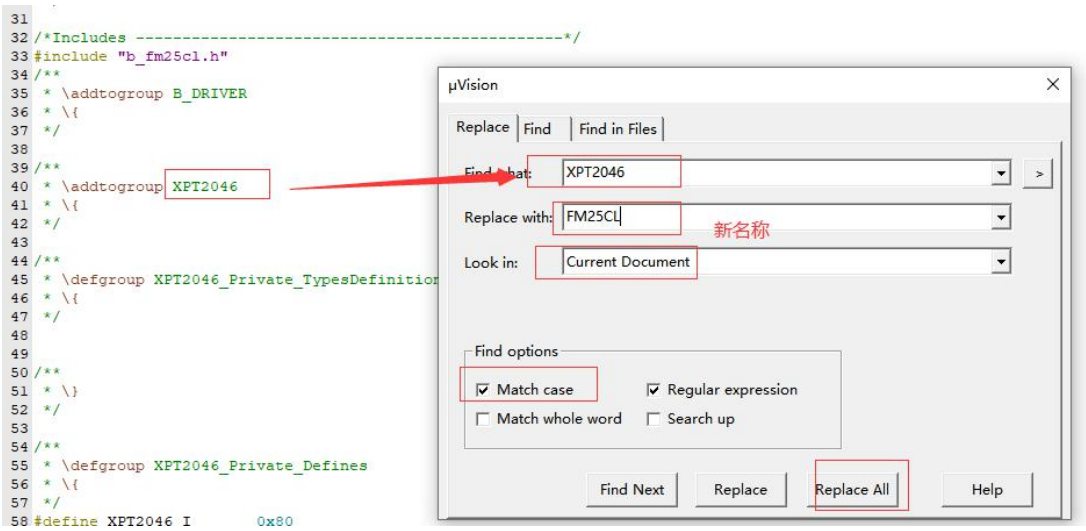
图 2-1 新建文件

2.3. 增加注释

将 b_xpt2046.c .h 内容分别拷贝过来（主要原因是这两个文件内容少，方便删除）。删除原有代码，保留注释部分。

修改如下几处注释：

1. 文件头部的文件信息，文件名、版本、日期、作者
2. 在.c 和.h 里面执行替换操作，如下图所示



3. 修改定义硬件接口的结构体类型

```
/**
 * \defgroup FM25CL_Exported_TypesDefinitions
 * \{
 */

typedef struct
{
    uint8_t (*pSPI_ReadWriteByte)(uint8_t);
    void (*pCS_Control)(uint8_t);
}bFM25CLPrivate_t;


typedef bDriverInterface_t bFM25CL_Driver_t;

/**
 * \}
 */
```

2.4. 完成功能部分代码

功能代码的模板是实现驱动接口里面的操作函数，open 和 close 放唤醒和睡眠的代码。Write/read 是读写数据，ctl 是对驱动做特别的操作，例如配置模式等。例如新增的驱动只有读写两个操作，那么初始化函数如下所示：

```
int bFM25CL_Init(bFM25CL_Driver_t *pdrv)
{
    pdrv->close = NULL;
    pdrv->read = _FM25_ReadBuff;
    pdrv->ctl = NULL;
    pdrv->open = NULL;
    pdrv->write = _FM25_WritBuff;
    return 0;
}
```

没有实现的部分用 NULL 补充。

2.5. 添加创建驱动的宏

在 b_driver.h 中添加新驱动的 h 文件，再写一个宏用于结合功能部分和硬件接口部分，如下所示：

```
34
35
36  #include "b_suart.h"
37  #include "b_w25x.h"
38  #include "b_xpt2046.h"
39  #include "b_ssd1289.h"
40  #include "b_f8l10d.h"
41  #include "b_fm25cl.h"
42
43
44  #define NEW_SUART_DRV(name, hal)        SUART_Driver_t name = {.init = SUART_Init,._private = &hal}
45
46  #define NEW_W25X_DRV(name, hal)         bW25X_Driver_t name = {.init = bW25X_Init,._private = &hal}
47
48  #define NEW_XPT2046_DRV(name, hal)      bXPT2046_Driver_t name = {.init = bXPT2046_Init,._private = &hal}
49
50  #define NEW_SSD1289_DRV(name, hal)      bSSD1289_Driver_t name = {.init = bSSD1289_Init,._private = &hal}
51
52  #define NEW_F8L10D_DRV(name, hal)       bF8L10D_Driver_t name = {.init = bF8L10D_Init,._private = &hal}
53
54  #define NEW_FM25CL_DRV(name, hal)       bF8L10D_Driver_t name = {.init = bFM25CL_Init,._private = &hal}
55
56
```

图 2-2 修改 b_driver.h

至此，新的驱动添加完成。

2.6. 收尾

自行测试成功后执行

```
git add .                    //添加改动
git commit -m “xxxxx”        //提交改动
git checkout dev             //切换回 dev 分支
git merge fm25c1             //合并代码
git push                     //推送至远程仓库
```