Структура проекта

Каждый раздел написан как отдельное занятие, хотя иногда и придётся использовать код из предыдущих разделов. Так или иначе, цель курса — создание рабочего устройства, поэтому нулевым шагом является — задать структуру проекта.

Согласно модульной парадигме программирования, стоит выделять в обособленные файлы все логически самостоятельные сущности. В некоторых случаях это может показаться бредово, выделять целый файл для одной функции длиной в одну строчку, но это плата за то, что бы структура оставалась понятной и прозрачной.

Каждому модулю будет соответствовать два файла — заголовочный и исходного кода. Например для микросхемы MAX7219:

```
// max7219.c
#include "max7266.h"
// code here

// max7219.h
#ifndef __MAX7219_H__
#define __MAX7219_H__
// declarations, macros here
#endif /* __MAX7219_H__ */
```

Создайте такие же пары файлов для каждой сущности. Потребуются еще два файла, заголовочный и файл исполняемого кода: `stm32f1xx_it.h` и `stm32f1xx_it.c`. В этих файлах будут описываться все обработчики прерываний, о которых мы еще поговорим.

> *Так как в курсе будет использовано аж четыре библиотеки (CMSIS, SPL, LL и HAL), то под каждую из них в репозитории заведена отдельная папка.*

```
main.c
------> stm32f10x.h
------> stm32f1xx_it.h
------> device_conf.h
------> main.h
------> button.h
------> buzzer.h
------> ds18b20.h
------> encoder.h
------> led.h
------> light_sensor.h
------> matrix.h
-----------------> max7219.h
------> rtc.h
------> state.h
------> utils.h
```

В файле `device_conf.h` можно хранить идентификатор устройства, версия прошивки, макросы `DEBUG` и `RELEASE`, а так же синонимы для всех необходимых констант (имена портов и т.д.).[1]

Кроме того, нужно создать дополнительный заголовочный файл `main.h` — это единственный заголовочный файл, который мы будем подключать к `main.c`. Вся работа с модулями будет идти через него.

```
#ifndef __MAIN_H__
#define __MAIN_H__

#include "stm32f10x.h"
#include "stm32f1xx_it.h"

#include "device_conf.h"

#include "button.h"
#include "buzzer.h"
#include "display.h"
#include "ds18b20.h"
#include "encoder.h"
#include "led.h"
#include "light_sensor.h"
#include "rtc.h"
#includue "state.h'
#include "utils.h"

void mcu_init(void);

#endif /* __MAIN_H__ */
```

Помимо этого, нам потребуется функция:

```
void mcu_init(void);
```

определение которой мы поместим непосредственно в `main.c`. Через нее будут вызываться все функции инициализации периферии микроконтроллера (в том числе и системы тактирования).

```
int main(void) {
    mcu_init();

    while (1) {
        // main loop
    }
}

void mcu_init(void) {
    // init code
}
```

Репозиторий на github: **CMSIS**.

Назад | **Оглавление** | **Дальше**

1. Полный пример `device_conf.h` в репозитории приведён только для SPL. ↩