

# ОПИСАНИЕ БИБЛИОТЕКИ STM32F2\_API

Автор: Дерябкин Вадим (Vadimatorik)

2017

# Глава 1

## ВВЕДЕНИЕ

В данном документе приводится исчерпывающее описание:

- философии библиотеки (логики построения и использования)
- соглашения о написании библиотеки (допустимые синтаксические приемы языка и общие правила написания кода)
- примеров использования библиотеки в реальных задачах

# Оглавление

<b>1</b>	<b>ВВЕДЕНИЕ</b>	<b>1</b>
<b>2</b>	<b>ФИЛОСОФИЯ БИБЛИОТЕКИ</b>	<b>3</b>
2.0.1	Общие сведения . . . . .	3
2.0.2	Краткий обзор реализации . . . . .	3

## Глава 2

# ФИЛОСОФИЯ БИБЛИОТЕКИ

### 2.0.1 Общие сведения

В основу библиотеки легли следующие постулаты:

1. Все, что можно вычислить на этапе компиляции - не должно вычисляться в реальном времени.
2. Между производительностью и расходом памяти выбор должен быть в сторону производительности.
3. Все, что может быть выполнено с помощью аппаратной периферии - не должно выполняться программно.
4. Библиотека должна иметь как можно больше средств гибкой настройки на этапе компиляции и по-минимуму - в реальном времени (в угоду производительности).
5. Работа программы должна быть по максимуму предсказуема еще на этапе компиляции. Отсюда следует, что все режимы работы периферии должны быть заданы статически.

### 2.0.2 Краткий обзор реализации

1. Библиотека написана на C++14.
2. Большую часть библиотеки составляют constexpr функции, которые обрабатывают заполненные пользователем структуры инициализации периферии на этапе компиляции и создают маски регистров для всевозможных, указанных в структуре инициализации, режимов. В реальном времени созданные из const constexpr структур инициализации глобальные объекты в коде пользователя оперируют созданными на этапе компиляции масками регистров для настройки и работы с периферийными блоками.

Этим достигается высокая производительность. Поскольку программе не нужно «собирать» маски регистров в реальном времени, как это сделано в HAL или SPL. Достаточно только применить маску.

3. Тот факт, что для инициализации глобальных объектов используются глобальные const constexpr структуры вовсе не означает, что данные структуры войдут в состав прошивки контроллера.

Яркий тому пример, объект класса global\_port (который будет рассмотрен в разделе). Он принимает в себя массив const constexpr pin\_config\_t структур, после чего private constexpr методы объекта класса global\_port их (структуры) анализируют и возвращают private global\_port\_msk\_reg\_struct структуру, которая будет private структурой глобального объекта класса global\_port.

Структуры `pin_config_t`, использовавшиеся для инициализации `private global_port_msk_reg` во `flash` загружены не будут, потому что в ходе работы программы обращений к ним не будет.