

# 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 [2.0.2](#)). 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` структуры, которая будет `private` структурой глобального объекта класса `global_port`.

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

4. Для работы с аппаратной частью контроллера используются объявленные в коде пользователя глобальные объекты. Все они должны быть объявлены как `const constexpr`. В качестве параметра(-ов) конструктора передается(-ются) указатель(-и) на `const constexpr` глобальную(-ые) структуру(-ы). Важно отметить следующее:

- В случае, если после анализа структур(-ы) инициализации они(-на) больше не требуется - линкер не включит эти(-у) структуры(-у) в состав выходного файла программы (о чем было сказано в пункте 3). Однако в случае, если используемая структура инициализации, возможно, будет использована во время выполнения программы, как, например, в классе `pin`, описанного в разделе 2.0.2, то она обязательно пойдет в состав выходной программы.

- Так как конструкторы классов используемых в коде пользователя объектов объявлены внутри класса как `constexpr`, то создание этих объектов, по сути, заключается в простом копировании в оперативную память их изменяемых данных. Никаких действий в реальном времени (за исключением копирования в оперативную память изменяемых в процессе работы данных объекта) не производится.

Объекты, классы которых имеют не `constexpr` конструктор (требующий вызова функции инициализации (конструктора) объекта перед вызовом `main` в реальном времени), не поддерживаются намеренно. Подробнее об этом рассказано в разделе, посвященному описанию логики работы `startup` файла (раздел 2.0.2).

- Из того, что все объекты объявлены как `const constexpr` следует, что у каждого глобального объекта, работающего с периферией в реальном времени, имеется метод начальной инициализации (и/или пере инициализации), вызов которого необходимо произвести из кода пользователя.

Это очень оправданно, когда требуется инициализировать объекты в определенном порядке в ходе выполнения программы, чего сложно достигнуть, когда объекты вызываются автоматически перед вызовом функции `main`. Именно это является причиной отказа от поддержки не `constexpr` конструкторов классов (вызов функций инициализации (конструкторов) которых, без применения дополнительных директив, производится в случайном порядке (нельзя гарантировать, инициализация какого объекта будет произведена раньше)).

- В случае, если `const constexpr` объект был объявлен глобально в коде пользователя, но обращений к нему не было на протяжении всей программы, он не будет добавлен в итоговый файл программы. Ситуация здесь аналогична ситуации с глобальными `const constexpr` структурами.