ВВЕДЕНИЕ В БИБЛИОТЕКУ STM32F2_API

Автор: Дерябкин Вадим (Vadimatorik)

2017

ВВЕДЕНИЕ

Данный документ создан с целью донести до пользователя:

- основные сведения о библиотеке (глава 1);
-
-

Оглавление

Глава 1

ЗНАКОМСТВО С БИБЛИОТЕКОЙ

1.1 Введение

Как известно, для микроконтроллеров существует множество различных библиотек, решающих широкий спектр задач. Однако большинство из них используют ресурсы микроконтроллера не экономно. Связано это с тем, что зачастую библиотеки пишутся под абстрактные микроконтроллеры и не учитывают индивидуальные особенности, имеющиеся у конкретных моделей. В своей библиотеке я поставил цель максимально эффективно использовать все доступные ресурсы конкретных серий микроконтроллеров.

1.2 Идеи, лежащие в основе библиотеки

На этапе проектирования библиотеки были отобраны следующие идеи, которые впоследствии легли в ее основу:

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 ещё на этапе компиляции. Отсюда следует, что все режимы работы периферии должны быть заданы статически.
6. Все используемые блоки периферии (как связанные с аппаратной периферией, так и являющиеся логической надстройкой) должны быть объявлены в коде пользователя в виде глобальных const constexpr объектов.

1.3 Краткий обзор реализации библиотеки

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 const constexpr объекты в коде пользователя оперируют созданными на этапе компиляции масками регистров для 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 1.3). 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_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 const constexpr объекты. 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 - компоновщик 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 1.3, 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 объекта (конструктора) перед вызовом main в реальном времени), **не поддерживаются намеренно**.
   - Из того, что все объекты объявлены как const constexpr следует, что у каждого глобального объекта, работающего в реальном времени, имеется метод начальной инициализации (и/или переинициализации), вызов которого необходимо произвести из кода пользователя.

     Это очень оправданно, когда требуется инициализировать объекты в определённом 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.