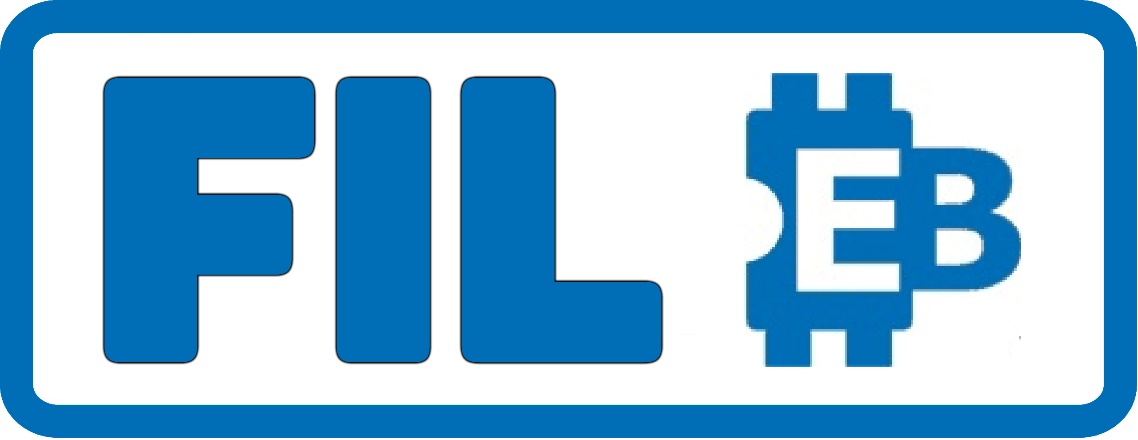
**STM32 Fast Initialization Library (FIL).**

Библиотека быстрой инициализации для микроконтроллеров STM32 на базе программной IDE EmBitz.

**Автор: Назаров А.А.**

****

**Содержание**

**Введение**……………………………………………………………….

1. **Структура библиотеки FIL**………………………………………….
   1. **Общее описание структуры**……………………………………..
   2. **Установка библиотеки FIL в текущий проект**…………………
2. **Начальный уровень разработчика \_Default**……………………

**2.1 Структура обучающей платформы**………………………………

**2.2 Список предоставляемых команд начального уровня**………...

**2.3 Примеры использования команд начального уровня**…………

1. **Продвинутый уровень разработчика \_Advanced**…………………

**Введение**

Популярность микроконтроллеров Atmega (Arduino series) в любительских проектах, а также некоторых профессиональных, обосновывается простотой использования и уменьшения, за счет этого, времени разработки полномасштабного и функционального алгоритма управления. Однако, для дальнейшего углубленного изучения принципа функционирования микроконтроллерных систем среда не предоставляет набор инструментария и функций. Пользователь погружаясь в дальнейшее изучение сталкивается с трудностями восприятия нового стиля написания программ через регистровые переменные, что увеличивает время обучения и проектирования алгоритмов.

Одним из популярных аналогов являются микроконтроллеры компании ST Microelectronics – STM8, STM32. Данные линейки контроллеров имеют обширную периферию, а доступ к изменению регистровых параметров позволяет разработчику настроить под себя систему управления, обмена данными и иными функциями. Общение через регистры низкоуровневым стилем программирования для начинающего компенсируется познанием базовых принципов и нюансов при инициализации и отладке программного кода. Данный подход не эффективен применительно к начинающим программировать данные микроконтроллеры. Поэтому сотрудниками Ресурсного центра робототехники (РЦР) Донского государственного технического университета (ДГТУ) была выдвинута идея создания полномасштабной библиотеки обучения и плавного перехода с высокого на низкий уровень проектирования алгоритмов.

Альфа версия библиотеки Fast Initialization Library (FIL) вышла в октябре 2021 года, имея в своем функционале инициализацию малого участка периферии. Поддержка продукта осуществляется и по сей день. За развитием проекта можно следить через платформу Github[1].

В данном документе приведены основные сведения для взаимодействия с библиотекой быстрой инициализации микроконтроллеров STM32. В основных разделах содержится информация по управляющим командам, а также примеры их использования. Приведена структура и принцип работы обучающей платформы, на основе которой демонстрируется функционал библиотеки. Документ дополняется по мере увеличения функционала библиотеки.

Все файлы и программный код библиотеки не претендуют на авторские права, любой желающий может использовать её в своих проектах. Приветствуется распространение и отзыв обратной связи о функциональности.

Для этого просьба обращаться по адресу - г.Ростов-на-Дону, пл.Гагарина 1, ДГТУ, 2 корпус, 2-205.

Также возможен вариант отправки письма на почту - [sasha.sanya.nazarov@yandex.ru](mailto:sasha.sanya.nazarov@yandex.ru) .

1. **Структура библиотеки FIL**

**1.1 Общее описание структуры**

Библиотека имеет разделение уровней доступа. Обосновывается это необходимостью обучения начинающих в программировании микроконтроллеров, поэтому имеется три уровня доступа пользователю:

\_Default – Режим новичка, который открывает доступ к элементарным методам (функциям), предоставляет возможность проектировать и компилировать алгоритмически построенный программный код, по минимуму затрагивая глубинную структуру регистровых списков. Участок создан посредством замены определений и действий библиотеки CMSIS на элементарные, понятные пользователю, макроопределения.

\_Advanced – Продвинутый режим. Открывает начальные функции библиотеки FIL. Используя данный уровень доступа, имеется возможность настраивать порты микроконтроллера, назначая им режим, скорость, установку подтягивающих резисторов, подключение альтернативных функций много других. Основная цель этого уровня – предоставить доступ к начальным функциям инициализации различных объектов периферии и считывания основных регистровых параметров, с целью фокусировки внимания на разработке проработанного алгоритма программного кода.

\_Developer – Уровень разработчика. Последний и самый приоритетный уровень доступа, открывающий все инструменты библиотеки FIL. Используя этот режим, разработчику предоставляется возможность контроля большинства параметров микроконтроллера, имеется доступ к методам гибкой настройки интерфейсов. Однако, уровень доступа сопровождается ростом входных переменных, которые должен указать разработчик.

Библиотека представлена следующими файлами и их назначением:

**filConfig.h** ̶ Главный файл настройки общей компиляции. Содержит блоки условной компиляции, предоставляющие защиту от неопределенности (не указан уровень доступа), переопределения (попытка указать все уровни доступа одновременно) и аппаратной несовместимости (подключенный контроллер не в списке поддерживаемых);

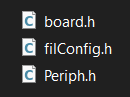
**Periph.h** ̶ Файл, содержащий функции инициализации и настройки периферийных элементов контроллера, функции считывания регистровых переменных;

**Board.h** ̶ Содержит в себе определения параметров платы обучения.

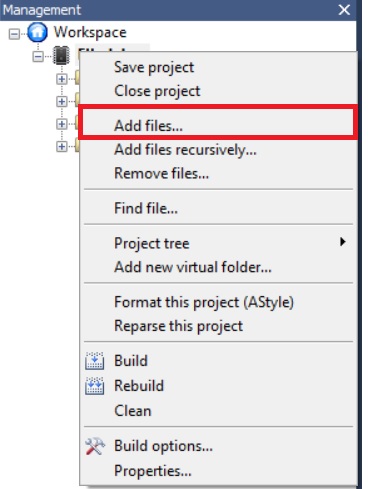
**1.2 Установка библиотеки FIL в текущий проект**

Для установки библиотечных файлов потребуется:

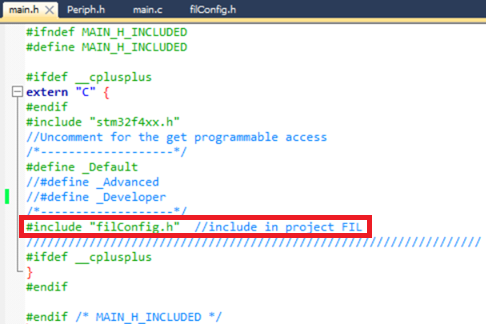
1. Файлы переместить в директорию с созданным проектом;



1. В используемой IDE добавить перенесенные файлы в текущий проект;



1. В заголовочном файле main.h добавить директиву обращения к библиотеке. Для этого добавьте строку #include <filConfig.h> как можно ниже, для предотвращения конфликта библиотеки с системными файлами проекта.

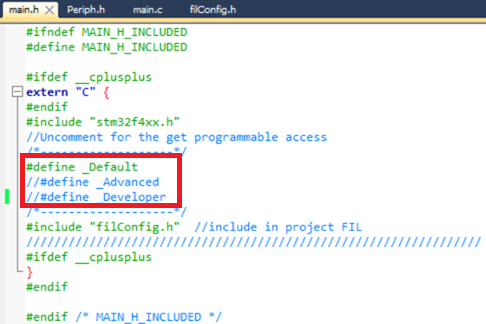


1. Библиотека защищена от неопределенности, поэтому необходимо указать уровень доступа в том же заголовочном файле main.h в соответствие с пунктом 1.1:

#define \_Default – для получения начального уровня;

#define \_Advanced – для получения продвинутого уровня;

#define \_Developer - для получения неограниченных прав разработчика;



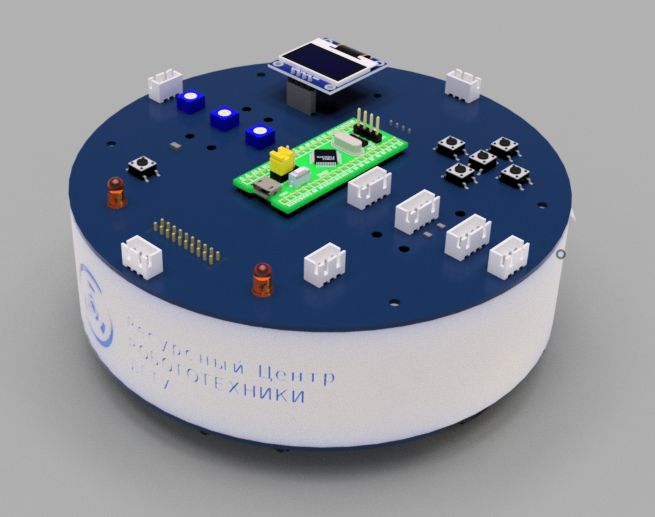
После проделанных пунктов библиотека готова к использованию.

1. **Начальный уровень разработчика \_Default**

Раздел предназначен для ознакомления только начавшим обучение программированию микроконтроллеров STM32. В сопровождении с этим документом предложены несколько типовых задач, решив которые, станет возможным переход на продвинутый уровень доступа.

* 1. **Структура обучающей платформы**

С целью разработки и последующей отладки программного кода РЦР ДГТУ предоставляет обучающую платформу, представляющую собой двухколесного робота. Обучаясь на данной платформе, разработчик сможет научиться программировать микроконтроллеры STM32 и продемонстрировать закрепленные знания выполнив несколько типовых заданий.



Данная платформа поддерживает несколько типовых функций: вывод информации на экран, программирование кнопок управления, светодиодов, управление исполнительными электроприводами.

* 1. **Список предоставляемых команд начального уровня**

На таблице 2.1 приведен общий список команд для разработчика, которые определены интуитивно понятными словесными конструкциями. Использование данного уровня не потребует углубленных знаний и используется для начального ввода в программирование микроконтроллеров.

Таблица 2.1 – список общих управляющих команд начального уровня.

|  |  |  |
| --- | --- | --- |
| **Название команды** | **Что делает** | **Какие параметры потребует** |
| ClocksInit | Инициализация периферии для работы с платой обучения | - |
| LedInit**\*** | Определение нужного количества светодиодов в обычном режиме | - |
| Set\_led(led)**\*\*** | Включает выбранный светодиод | led – номер светодиода, который необходимо включить |
| PWM\_init(a)**\*** | Определение нужного количества светодиодов в специальном режиме работы (ШИМ) | a – количество необходимых светодиодов для режима ШИМ |
| SetPWM(led,duty)**\*\*** | Установка заполнения на выбранный светодиод. Значение заполнения задается от 0 до 100% | led – номер необходимого светодиода;  duty – требуемое заполнение на выбранном светодиоде. |
| But\_init(buttons) | Определение нужного количества кнопок | Buttons – количество необходимых кнопок (максимум 2). |
| ADC\_init | Инициализация аналого-цифрового преобразователя (АЦП) для работы с переменным резистором и датчиком линии | - |
| **\*** ̶ Имеется защита от переопределения | | |
| **\*\* ̶** Имеется защита от неверного указания параметров | | |

Приведенного списка команд достаточно для выполнения практических задач начального уровня.

* 1. **Примеры использования команд начального уровня**

При разработке программного кода начинающему следует использовать правильный порядок написания программного кода. Для этого рассмотрим примитивный пример решения практической задачи:

*Используя инструменты начального уровня, напишите алгоритм включения светодиода №3.*

Для решения этой задачи был написан код приведенный ниже.

#include “main.h”

int main(void){

Clocks\_init;

Led\_init(3);

Set\_led(3);

While(1) {}

Как видно из примера, первой и главной командой является строка Clocks\_init;, которая разблокирует дальнейшую настройку, если строку не включать то последующие действия не окажут никакого эффекта на микроконтроллер. Далее следует функция определения светодиодов Led\_init(3);. Так как необходим светодиод №3, то вынужденная мера — это определение 3 светодиодов, чтобы определить нужный. Функция имеет защиту от переопределения и неопределенности аргументов (см. примечание 2.1). Завершающей выполнение задачи функция Set\_led(3); включает светодиод №3, так как указанное значение в скобках – число 3. Команда Reset\_led(3); является противоположностью команды Set\_led(3);, для решения текущей задачи не подходит, потому что выключает светодиод №3, когда нужно его включить.

Задача осложняется, когда необходимо не простое включение и выключение светодиода, а периодичность включения и выключения с привязкой ко времени. Следующий пример демонстрирует работу со специальным режимом ШИМ светодиода:

*С помощью инструментов начального уровня, включите светодиод №3 с периодом мерцания 0,5 секунды.*

#include “main.h”

int main(void){

Clocks\_init;

PWM\_init(3);

SetPWM(3,0.5);

While(1) {}

Нетрудно определить, что данный пример отличается от предыдущего, появлением двух новых команд. После этапа инициализации функция PWM\_init(3);, настраивает 3 светодиода на работу в специальном режиме (см. примечание 2.1). После настройки светодиодов необходимо задать 3 светодиоду частоту мерцания пол секунды (0,5 сек) с помощью функции SetPWM(3,0.5);. Значение 3 соответствует необходимому сейчас светодиоду, а число 0.5 – заполнению. Мерцание достигается за счет значения заполнения – процент включения/выключения светодиода за одну единицу времени, в данном случае, секунду. Поэтому, для того, чтобы инициировать мерцание в 0,5 секунды необходимо установить заполнение 50%, то есть значение 0.5, обязательно с точкой, а не запятой.

*Примечание 2.1 – Функции Led\_init()* *и PWM\_init() защищены от переопределения. Это значит, что комбинированное использование одной и другой в программе приведет к тому, что первая выполняется и действует ровно до того момента, пока не определится вторая.*

1. **Продвинутый уровень разработчика \_Advanced**

В данном разделе приложено описание продвинутого уровня доступа, открывающий возможность частичного использования библиотеки FIL. Основной отличительной чертой данной уровня от начального является возможность инициализации большей части интерфейсов с помощью упрощенно задаваемых команд. Однако использование продвинутого уровня потребует от разработчика задавать больше входных параметров, производить базовые расчеты для инициализации необходимой периферии.

* 1. **Дополнения к базовой обучающей платформе для продвинутого разработчика**

**-**

**3.2 Список предоставляемых команд продвинутого уровня разработчика**

**3.2.1 Порты ввода-вывода (GPIO)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название команды** | | **Что делает** | **Какие параметры потребует** | |
| pin\_id(PIN\_PORT,PIN) | | Инициализация определенного порта микроконтроллера для дальнейшей настройки в других командах. Является базовой командой. | PIN\_PORT – группа, в которой находится нужный порт;  PIN – номер необходимого порта в группе. | |
| set\_x(pin,x),  clear\_x(pin) | mode | Назначение/сброс режима порта | pin – необходимый порт |  |
| type | Назначение/сброс типа подключения порта |  |
| speed | Назначение/сброс скорости работы порта |  |
| pull | Назначение/сброс подтяжек порта |  |
| af | Назначение/сброс альтернативной функции порта |  |
|  |  |  |  | |
|  | |  |  | |