**

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «Прикладное программирование»**

**Тема: «Безопасный заряд Li-Ion аккумулятора»**

Выполнил студент

Живин Радамир Александрович

Группа П3-16

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Королев, 2019

Оглавление

[**Задание на выполнение курсового проекта** 3](#_Toc40806822)

[**Введение** 4](#_Toc40806823)

[**1.** **Теоретическая часть** 5](#_Toc40806824)

[**1.1.** **Li-Ion аккумуляторы** 5](#_Toc40806825)

[**3.** **Проектная часть** 10](#_Toc40806826)

[**3.1.** **Описание инструментов разработки программы** 10](#_Toc40806827)

[**3.2.** **Описание разработанного приложения** 12](#_Toc40806828)

[**3.3.** **Описание основных функций, используемых в программе** 12](#_Toc40806829)

[**3.3.1.** **Список файлов** 12](#_Toc40806830)

[3.3.2. Функция тела программы (файл main.c) 15](#_Toc40806831)

[3.3.3. Конфигурация таймера (файл main.c) 18](#_Toc40806832)

[3.3.4. Обработка прерывания таймера (файл main.c) 19](#_Toc40806833)

[3.3.5. Обработка прерывания АЦП (файл itogwin.cpp) 19](#_Toc40806834)

[Заключение. 21](#_Toc40806835)

[Список источников. 22](#_Toc40806836)

# **Задание на выполнение курсового проекта**

**Краткое описание программы:**

Разработать программу безопасного заряда Li-Ion аккумулятора для микроконтроллера STM32F051R8Tx с использованием языка программирования C и среды разработки IAR Embedded Workbench for ARM (EWARM) и конфигуратора STM32Cube MX.

**Полное описание задачи:**

1. Сконфигурировать микроконтроллер.

Указать пин входных данных и пин выходных данных, указать пины входных данных для АЦП, указать пин выходных данных ЦАП, создать и настроить таймер.

1. Программно описать процесс заряда.

Во время заряда необходимо контроллировать уровни тока и напряжения, чтобы плавный заряд осуществлялся эффективно и безопасно.

1. Описать системные прерывания.

Прерывания вызываются модулями микроконтроллера. АЦП вызывает прерывание при окончании считывания данных с входных пинов, на которые подаётся ток и напряжение. Таймер вызывает прерывание при прохождении итерации.

**Входные данные:**

На вход подаётся ток и напряжение аккумулятора.

**Выходные данные:**

Выходные данные формируются ЦАП и подают нужный ток на аккумулятор.

**Введение**.

В процессе используется IAR Embedded Workbench for ARM — среда разработки для встроенных систем на архитектуре ARM. Программа должна конфигурировать микроконтроллер для его дальнейшей эксплуатации в качестве зарядного устройства Li-Ion аккумуляторов, для этого необходимо соблюсти точность в рассчётах получаемых и подаваемых значений. В первой части будут рассмотрены: предметная область, история, характеристики и принцип работы Li-Ion аккумуляторов. Во второй части будут рассмотренны характеристики и области применения микроконтроллера STM32F051R8Tx. В третьей части будут рассмотрены инструменты и библиотеки, которые были задействованы при разработке программной части проекта, структура программной части и листинги ключевых частей программных модулей.

Для осуществления обозначенных целей служат следующие задачи:

* 1. Разработка программы и её функций
  2. Написание кода программы

# **Теоретическая часть**

# **Li-Ion аккумуляторы**

**Литий-ионный аккумулятор (Li-ion)** — тип электрического аккумулятора, который широко распространён в современной бытовой электронной технике и находит своё применение в качестве источника энергии в электромобилях и накопителях энергии в энергетических системах. Это самый популярный тип аккумуляторов в таких устройствах как сотовые телефоны, ноутбуки, цифровые фотоаппараты, видеокамеры и электромобили.

Рисунок 1. Li-Ion аккумулятор цилиндрический, типоразмера 18650

Рисунок 2. Li-Ion аккумулятор сотового телефона Siemens, призматический

* 1. **История.**

Впервые принципиальная возможность создания литиевых аккумуляторов на основе способности дисульфид титана или дисульфид молибдена включать в себя ионы лития при разряде аккумулятора и экстрагировать их при зарядке была показана в 1970 году Майклом Стэнли Уиттингемом. Существенным недостатком таких аккумуляторов являлось низкое напряжение - 2,3В и высокая пожароопасность вследствие образования дендритов металлического лития, замыкающих электроды.

Позднее Дж. Гуденафом были синтезированы другие материалы для катода литиевого аккумулятора - кобальтит лития LixCoO2 (1980 год), феррофосфат лития LiFePO4 (1996 год). Преимуществом таких аккумуляторов является более высокое напряжение - около 4В.

Современный вариант литий-ионного аккумулятора с анодом из графита и катодом из кобальтита лития изобрёл в 1991 году Акира Ёсино. Первый литий-ионный аккумулятор по его патенту выпустила корпорация Sony в 1991 году.

В настоящее время ведутся исследования по поиску материалов на основе кремния и фосфора, обеспечивающих повышенную емкость интеркалирования ионов лития и по замене ионов лития на ионы натрия.

Нобелевская премия по химии 2019 года была вручена троим перечисленным выше учёным "За создание литий-ионных батарей".

* 1. **Характеристики.**

Характеристики литий-ионных аккумуляторов зависят от химического состава составляющих компонентов и варьируются в следующих пределах:

* напряжение единичного элемента:
  + максимальное: 4,2 В (или 4,35/4,40 В для высоковольтных);
  + минимальное: 2,5 В (или 2,8/3,0 В для высоковольтных);
* удельная энергоёмкость: 110…270 Вт\*ч/кг;
* внутреннее сопротивление: 4…15 мОм\*Ач;
* число циклов заряд-разряд до снижения ёмкости до 80%: 600;
* время быстрого заряда: 1 час;
* саморазряд зависит от температуры хранения и степени заряда. При температуре 25°C и заряде 100% ≈ 1,6% в месяц;
* ток нагрузки относительно ёмкости **С** представленной в А\*ч:
  + постоянный: до 5**С**;
  + импульсный: до 50**С**;
  + оптимальный: до 1**С**;
* диапазон рабочих температур: от -20°C до +60°C (оптимальная +23°C)
  1. **Устройство.**

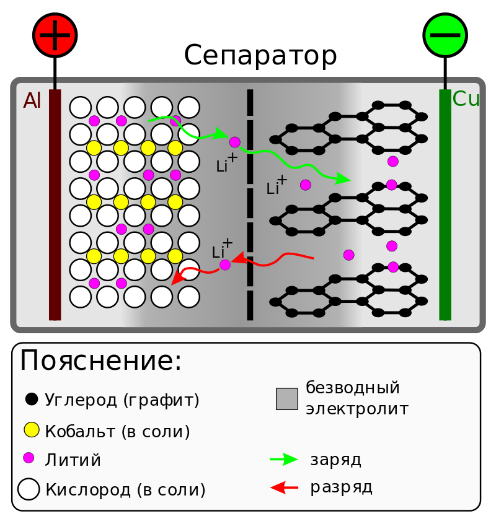
Литий-ионный аккумулятор состоит из электродов (катодного материала на алюминиевой фольге и анодного материала на медной фольге), разделённых пористым сепаратором, пропитанным электролитом. Пакет электродов помещён в герметичный корпус, катоды и аноды подсоединены к клеммам-токосъёмникам. Корпус иногда оснащают предохранительным клапаном, сбрасывающим внутреннее давление при аварийных ситуациях или нарушениях условий эксплуатации. Литий-ионные аккумуляторы различаются по типу используемого катодного материала. Переносчиком заряда в литий-ионном аккумуляторе является положительно заряженный ион лития, который имеет способность внедряться (интеркалироваться) в кристаллическую решётку других материалов (например, в графит, оксиды и соли металлов) с образованием химической связи, например: в графит с образованием LiC6, оксиды (LiMnO2) и соли (LiMnRON) металлов.

Рисунок 3. Li-Ion аккумулятор. Схема работы

Первоначально в качестве отрицательных пластин применялся металлический литий, затем — каменноугольный кокс. В дальнейшем стал применяться графит. Применение оксидов кобальта позволяет аккумуляторам работать при значительно более низких температурах, повышает количество циклов разряда/заряда одного аккумулятора. Распространение литий-железо-фосфатных аккумуляторов обусловлено их относительно низкой стоимостью. Литий-ионные аккумуляторы применяются в комплекте с системой контроля и управления — СКУ или BMS (battery management system), — и специальным устройством заряда/разряда.

В настоящее время в массовом производстве литий-ионных аккумуляторов используются три класса катодных материалов:

* кобальтат лития LiCoO2 и твёрдые растворы на основе изоструктурного ему никелата лития
* литий-марганцевая шпинель LiMn2O4
* литий-феррофосфат LiFePO4.

Электрохимические схемы литий-ионных аккумуляторов:

* литий-кобальтовые LiCoO2 + 6C → Li1-xCoO2 + LiC6
* литий-ферро-фосфатные LiFePO4 + 6C → Li1-xFePO4 + LiC6

Благодаря низкому саморазряду и большому количеству циклов заряда/разряда, Li-ion-аккумуляторы наиболее предпочтительны для применения в альтернативной энергетике. При этом, помимо системы СКУ они укомплектовываются инверторами (преобразователи напряжения).

* 1. **Принцип работы**

Суть работы батареи на литии состоит в обеспечении оптимальных условий для перемещения ионов металла внутри системы (а точнее – между разнозаряженными электродами).

Вот как работает литий-ионный аккумулятор в стандартной ситуации:

* когда на электроды подается напряжение определенной величины, это стимулирует ионы Li переходить из литиевого катода в угольный анод. Этот процесс сопровождается окислительной реакцией.
* когда же в систему подается нагрузка, это заставляет ионы металла передвигаться в обратном направлении.

Однако, ввиду того, что при разряжении батареи, отрицательный электрод возвращается в норму не до конца, а параллельно с этим еще и накапливаются продукты окисления, то АКБ медленно, но уверенно лишается части своей емкости. В момент, когда фиксируется потеря 30% объема батареи, говорят о завершении ее жизненного цикла.

* 1. **Разновидности литиевых аккумуляторов**

В современном мире существует огромное число модификаций литиевых аккумуляторных батарей. На данный момент наибольшее распространение в производстве получили только некоторые из них:

* литий-железо-фосфатные, славящиеся износоустойчивостью, высокой термостабильностью, эксплуатационной безопасностью и длительным периодом работы;
* литий-кобальтовые, выделяющиеся на фоне аналогов показательной удельной энергоемкостью, но малой термостабильностью и непродолжительным жизненным циклом;
* литий-марганцевые, чье главное преимущество заключается в умеренной удельной энергоемкости, хотя и при низком сопротивлении;
* литий-никель-кобальт-алюминий-оксидные с хорошей плотностью и энергоемкостью, многообещающей продолжительностью функционирования;
* литий-титанатные, обеспечивающие быструю зарядку, хорошую производительность, способность не терять свою емкость при критических температурах;
* литий-никель-марганец-кобальт-оксидные дают низкое внутреннее сопротивление, высокую удельную емкость.

Применяются аккумуляторы при создании разной продукции: Li-Co – в ноутбуках, смартфонах, видео- и фотокамерах, Li-Mn – в медтехнике и специнструментах, LiNiMnCoO2 – в электромобилях, электровелосипедах, телекоммуникациях, электростанциях, системах безопасности; LiFePO4 – в оборудовании, рассчитанном на большой ток нагрузки, LiTi – в уличном освещении, источниках бесперебойного питания, электротранспорте; LiNiCoAlO2 – в силовых агрегатах, медразработках.

Усовершенствованной версией литиевых батарей считаются полимерные АКБ, использующие гелеобразный, сухой или выполненный из полимерной матрицы, электролит. В таких устройствах электролит помещается на полимерную пленку, обеспечивающую хороший обмен ионами. Подобная конструкция обуславливает микроразмеры ячеек, безопасность их эксплуатации и простоту изготовления

* 1. **Способы заряда Li-Ion аккумуляторов**

Для заряда Li-ion аккумуляторов используется метод «постоянное напряжение / постоянный ток», суть которого заключается в ограничении напряжения на аккумуляторе. В этом он подобен методу заряда свинцово-кислотных аккумуляторов (SLA). Основные отличия заключаются в том, что для Li-ion аккумуляторов — выше напряжение на элемент (номинальное напряжение элемента 3,6В против 2В для SLA), более жесткий допуск на это напряжение (±0,05 В) и отсутствие медленного подзаряда по окончании полного заряда.

Для примера приведем требования и рекомендации по заряду и разряду литий-ионных аккумуляторов фирмы Panasonic:

* максимальное напряжение заряда 4,2 или 4,1 вольта в зависимости от модели аккумулятора;
* напряжение окончания разряда 3,0 вольта;
* рекомендуемый ток заряда 0,7 С, ток разряда (нагрузки) — 1 С и меньше;
* если напряжение на аккумуляторе менее 2,9 вольта, то рекомендуемый ток заряда 0,1 С;
* глубокий разряд может привести к повреждению аккумулятора (т. е. должно соблюдаться общее правило — Li-ion аккумуляторы любят скорее находиться в заряженном состоянии, чем в разряженном, и заряжать их можно в любое время, не дожидаясь разряда);
* по мере приближения напряжения на аккумуляторе к максимальному значению, ток заряда уменьшается. Окончание разряда должно происходить при уменьшении тока заряда до (0,1 … 0,07) С в зависимости от модели аккумулятора. После окончания заряда ток заряда прекращается полностью.
* диапазон температур при заряде от 0 до 45 градусов Цельсия, при разряде от минус 10 до 60 градусов Цельсия.

Приведенные выше данные могут отличаться в ту или иную сторону для аккумуляторов других производителей.

В то время как для SLA аккумуляторов допустима некоторая гибкость в установке значения напряжения прекращения заряда, для Li-ion аккумуляторов изготовители очень строго подходят к выбору этого напряжения. Порог напряжения прекращения заряда для Li-ion аккумуляторов 4,10 В или 4,20 В, допуск на установку для обоих типов ±0,05 В на элемент. Для вновь разрабатываемых Li-ion аккумуляторов, вероятно, будут определены другие значения этого напряжения. Следовательно, зарядные устройства для них должны быть адаптированы к требуемому напряжению заряда.

Более высокое значение порога напряжения обеспечивает и большее значение емкости, поэтому в интересах изготовителя выбрать максимально возможный порог напряжения без нарушения безопасности. Однако на величину этого порога влияет температура аккумулятора, и его устанавливают достаточно низким для того, чтобы допустить повышенную температуру при заряде.

В зарядных устройствах и анализаторах аккумуляторов, которые позволяют изменять значение этого порога напряжения, его правильная установка должна соблюдаться при обслуживании любых аккумуляторов Li-ion типа. Однако большинство изготовителей не обозначают тип Li-ion аккумулятора и напряжения окончания заряда. И, если напряжение установлено неправильно, то аккумулятор с более высоким напряжением выдаст более низкое значение емкости, а аккумулятор с более низким — будет немного перезаряжен. При умеренной температуре повреждения аккумуляторов не происходит.

Именно в этом, как правило, и заключается причина того, что аккумулятор, заряженный, например, в «родном» телефоне, работает меньшее или большее время, чем этот же аккумулятор, заряженный в настольном зарядном устройстве неизвестного производителя.

Повышение температуры аккумулятора при заряде незначительно (от 2 до 8 градусов в зависимости от типа и производителя)

Вмешательство потребителя в любое Li-ion зарядное устройство не рекомендуется.

Медленный подзаряд по окончании заряда, характерный для аккумуляторов на основе никеля, не применяется, потому что Li-ion аккумулятор не терпит перезаряда. Медленный заряд может вызвать металлизацию лития и привести к разрушению элемента. Вместо этого время от времени для компенсации маленького саморазряда аккумулятора из-за небольшого тока потребления устройством защиты может применяться кратковременный заряд.

Li-ion аккумуляторы содержат несколько встроенных устройств защиты: плавкий предохранитель, термопредохранитель и внутреннюю схему управления, которая отключает аккумулятор в нижней и верхней точках напряжения разряда и заряда.

1. **Микроконтроллеры STM32**
   1. **Архитектура ARM (Advanced RISC Machine)**

Архитектура ARM — семейство лицензируемых [32-битных](https://ru.wikipedia.org/wiki/32_%D0%B1%D0%B8%D1%82) и [64-битных](https://ru.wikipedia.org/wiki/64_%D0%B1%D0%B8%D1%82) [микропроцессорных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) ядер разработки компании [ARM Limited](https://ru.wikipedia.org/wiki/ARM_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)).

Среди [лицензиатов](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F) — [AMD](https://ru.wikipedia.org/wiki/Advanced_Micro_Devices), [Apple](https://ru.wikipedia.org/wiki/Apple), [Analog Devices](https://ru.wikipedia.org/wiki/Analog_Devices), [Atmel](https://ru.wikipedia.org/wiki/Atmel), [Xilinx](https://ru.wikipedia.org/wiki/Xilinx), [Cirrus Logic](https://ru.wikipedia.org/w/index.php?title=Cirrus_Logic&action=edit&redlink=1), [Intel](https://ru.wikipedia.org/wiki/Intel) (до 27 июня 2006 года), [Marvell](https://ru.wikipedia.org/wiki/Marvell_Technology_Group), [NXP](https://ru.wikipedia.org/wiki/NXP_Semiconductors), [**STMicroelectronics**](https://ru.wikipedia.org/wiki/STMicroelectronics), [Samsung](https://ru.wikipedia.org/wiki/Samsung), [LG](https://ru.wikipedia.org/wiki/LG_Group), [MediaTek](https://ru.wikipedia.org/wiki/MediaTek), [Qualcomm](https://ru.wikipedia.org/wiki/Qualcomm), [Sony](https://ru.wikipedia.org/wiki/Sony), [Texas Instruments](https://ru.wikipedia.org/wiki/Texas_Instruments), [Nvidia](https://ru.wikipedia.org/wiki/Nvidia), [Freescale](https://ru.wikipedia.org/wiki/Freescale_Semiconductor), [Миландр](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BB%D0%B0%D0%BD%D0%B4%D1%80), ЭЛВИС, [HiSilicon](https://ru.wikipedia.org/wiki/HiSilicon_Technologies).

Значимые семейства процессоров: [ARM7](https://ru.wikipedia.org/wiki/ARM7" \o "ARM7), [ARM9](https://ru.wikipedia.org/wiki/ARM9), [ARM11](https://ru.wikipedia.org/wiki/ARM11) и [Cortex](https://ru.wikipedia.org/wiki/ARM_Cortex).

Многие лицензиаты делают собственные версии ядер на базе ARM: [DEC](https://ru.wikipedia.org/wiki/DEC" \o "DEC) [StrongARM](https://ru.wikipedia.org/wiki/StrongARM), [Freescale](https://ru.wikipedia.org/wiki/Freescale) i.MX, [Intel](https://ru.wikipedia.org/wiki/Intel) [XScale](https://ru.wikipedia.org/wiki/Intel_XScale), [NVIDIA Tegra](https://ru.wikipedia.org/wiki/NVIDIA_Tegra), [ST-Ericsson](https://ru.wikipedia.org/wiki/ST-Ericsson) [Nomadik](https://ru.wikipedia.org/w/index.php?title=Nomadik&action=edit&redlink=1), [Krait](https://ru.wikipedia.org/wiki/Krait_(CPU)) и [Kryo](https://ru.wikipedia.org/w/index.php?title=Kryo&action=edit&redlink=1) в [Qualcomm](https://ru.wikipedia.org/wiki/Qualcomm) [Snapdragon](https://ru.wikipedia.org/wiki/Snapdragon), [Texas Instruments](https://ru.wikipedia.org/wiki/Texas_Instruments) [OMAP](https://ru.wikipedia.org/wiki/Texas_Instruments_OMAP), [Samsung](https://ru.wikipedia.org/wiki/Samsung) [Hummingbird](https://ru.wikipedia.org/wiki/Exynos), [LG H13](https://ru.wikipedia.org/w/index.php?title=LG_H13&action=edit&redlink=1), [Apple](https://ru.wikipedia.org/wiki/Apple) [A6](https://ru.wikipedia.org/wiki/Apple_Ax) и [HiSilicon K3](https://ru.wikipedia.org/wiki/HiSilicon_K3).

* + 1. **Процессоры ARM**

В настоящее время значимыми являются несколько семейств процессоров ARM:

* ARM7 (с тактовой частотой до 60-72 МГц), предназначенные, например, для недорогих мобильных телефонов и встраиваемых решений средней производительности. В настоящее время активно вытесняется новым семейством Cortex.
* ARM9, ARM11 (с частотами до 1 ГГц) для более мощных телефонов, карманных компьютеров и встраиваемых решений высокой производительности.
* Cortex A — новое семейство процессоров на смену ARM9 и ARM11.
* Cortex M — новое семейство процессоров на смену ARM7, также призванное занять новую для ARM нишу встраиваемых решений низкой производительности. В семействе присутствуют четыре значимых ядра:
* Cortex-M0, Cortex-M0+ (более энергоэффективное) и Cortex-M1 (оптимизировано для применения в ПЛИС) с архитектурой ARMv6-M;
* Cortex-M3 с архитектурой ARMv7-M;
* Cortex-M4 (добавлены SIMD-инструкции, опционально FPU) и Cortex-M7 (FPU с поддержкой чисел одинарной и двойной точности) с архитектурой ARMv7E-M;
* Cortex-M23 и Cortex-M33 с архитектурой ARMv8-M ARMv8-M .

В 2010 году производитель анонсировал процессоры Cortex-A15 под кодовым названием Eagle, ARM утверждает, что ядро Cortex A15 на 40 процентов производительнее на той же частоте, чем ядро Cortex-A9 при одинаковом числе ядер на чипе. Изделие, изготовленное по 28-нанометровому техпроцессу, имеет 4 ядра, может функционировать на частоте до 2,5 ГГц и будет поддерживаться многими современными операционными системами.

Популярное семейство микропроцессоров xScale фирмы Marvell (до 27 июня 2007 года — Intel), в действительности является расширением архитектуры ARM9, дополненной набором инструкций Wireless MMX, специально разработанных фирмой Intel для поддержки мультимедийных приложений.

* + 1. **Архитектура**

Уже давно существует справочное руководство по архитектуре ARM, которое разграничивает все типы интерфейсов, которые поддерживает ARM, так как детали реализации каждого типа процессора могут различаться. Архитектура развивалась с течением времени и, начиная с ARMv7, были определены 3 профиля:

* A (application) — для устройств, требующих высокой производительности (смартфоны, планшеты);
* R (real time) — для приложений, работающих в реальном времени;
* M (microcontroller) — для микроконтроллеров и недорогих встраиваемых устройств.

Профили могут поддерживать меньшее количество команд (команды определенного типа).

**Режимы**

Процессор может находиться в одном из следующих операционных режимов:

* User mode — обычный режим выполнения программ. В этом режиме выполняется большинство программ.
* Fast Interrupt (FIQ) — режим быстрого прерывания (меньшее время срабатывания).
* Interrupt (IRQ) — основной режим прерывания.
* System mode — защищённый режим для использования операционной системой.
* Abort mode — режим, в который процессор переходит при возникновении ошибки доступа к памяти (доступ к данным или к инструкции на этапе prefetch конвейера).
* Supervisor mode — привилегированный пользовательский режим.
* Undefined mode — режим, в который процессор входит при попытке выполнить неизвестную ему инструкцию.

Переключение режима процессора происходит при возникновении соответствующего исключения или же модификацией регистра статуса.

**Функции RISC**

Архитектура ARM обладает следующими особенностями RISC:

* Архитектура загрузки/хранения
* Нет поддержки нелинейного (не выровненного по словам) доступа к памяти (теперь поддерживается в процессорах ARMv6, за некоторыми исключениями, и полностью в ARMv7)
* Равномерный 16х32-битный регистровый файл
* Фиксированная длина команд (32 бита) для упрощения декодирования за счет снижения плотности кода. Позднее режим Thumb повысил плотность кода.
* Одноцикловое исполнение
* Чтобы компенсировать простой дизайн, в сравнении с современными процессорами вроде Intel 80286 или Motorola 68020 были использованы некоторые особенности дизайна:
* Арифметические инструкции заменяют условные коды, только когда это необходимо
* 32-битное многорегистровое циклическое сдвиговое устройство, которое может быть использовано без потерь производительности в большинстве арифметических инструкций и адресных расчетов.
* Мощные индексированные адресные режимы
* Регистр ссылок для быстрого вызова функций листьев
* Простые, но быстрые, с двумя уровнями приоритетов подсистемы прерываний с включенными банками регистров.

**Условное исполнение**

Одним из существенных отличий архитектуры ARM (изначальная архитектура) от других архитектур ЦПУ является так называемая предикация — возможность условного исполнения команд. Под «условным исполнением» здесь понимается то, что команда будет выполнена или проигнорирована в зависимости от текущего состояния флагов состояния процессора. В Thumb и Arm 64 предикация не используется — в первом режиме для неё нет места в команде (всего 16 бит), а во втором предикация бессмысленна и сложна для реализации на суперскалярных архитектурах.

В то время как для других архитектур таким свойством, как правило, обладают только команды условных переходов, в архитектуру ARM была заложена возможность условного исполнения практически любой команды. Это было достигнуто добавлением в коды их инструкций особого 4-битового поля (предиката). Одно из его значений зарезервировано на то, что инструкция должна быть выполнена безусловно, а остальные кодируют то или иное сочетание условий (флагов). С одной стороны, с учётом ограниченности общей длины инструкции, это сократило число битов, доступных для кодирования смещения в командах обращения к памяти, но с другой — позволило избавляться от инструкций ветвления при генерации кода для небольших if-блоков.

**Процесс запуска ОС на ARM-машинах**

После включения системы на базе ARM-процессора из ROM-памяти загружается начальный загрузчик и адрес его точки входа. Начальный загрузчик проводит предварительную инициализацию системы, исполняя тем самым ту же роль, которую исполняет BIOS на системах x86, после чего может загрузить либо системный загрузчик, либо напрямую ОС.

Единого стандарта на начальный загрузчик не существует. Может применяться загрузчик U-Boot, а для 64-х разрядной ARMv8 часто используется UEFI. Во многих применениях используются собственные загрузчики.

**ОС, поддерживающие ARM**

Архитектура ARM поддерживается множеством операционных систем. Наиболее широко используемые: Linux (в том числе Android), iOS, Windows Phone. Подробнее: Список операционных систем с поддержкой процессоров ARM.

Работать на системах с ARM-процессором могут различные Unix и Unix-подобные ОС: Linux (многие дистрибутивы), iOS, Android, BSD (FreeBSD, NetBSD, OpenBSD), QNX, Plan 9, Inferno, OpenSolaris (2008—2009[54]), Firefox OS.

Также на платформе запускаются отдельные варианты семейства Windows: Windows CE, Windows Phone, Windows RT, Windows 10 (Только на Raspberry Pi или Lumia 950 / 950 XL)

Кроме того, ARM поддерживают: A2, FreeRTOS, Nucleus, Symbian OS, RISC OS, RISC iX.

* 1. **Обзор микроконтроллеров STM32**

**STM32** — семейство микроконтроллеров, основанных на [32-битных](https://ru.wikipedia.org/wiki/32_%D0%B1%D0%B8%D1%82%D0%B0) ядрах ARM [Cortex-M7F](https://ru.wikipedia.org/w/index.php?title=Cortex-M7F&action=edit&redlink=1), [Cortex-M4F](https://ru.wikipedia.org/w/index.php?title=Cortex-M4F&action=edit&redlink=1), [Cortex-M3](https://ru.wikipedia.org/w/index.php?title=Cortex-M3&action=edit&redlink=1), [Cortex-M0+](https://ru.wikipedia.org/w/index.php?title=Cortex-M0%2B&action=edit&redlink=1) или [Cortex-M0](https://ru.wikipedia.org/w/index.php?title=Cortex-M0&action=edit&redlink=1) с [сокращённым набором инструкций](https://ru.wikipedia.org/wiki/RISC). [STMicroelectronics](https://ru.wikipedia.org/wiki/STMicroelectronics" \o "STMicroelectronics) (ST) имеет лицензию на [IP процессора](https://ru.wikipedia.org/wiki/IP-cores) ARM от [ARM Holdings](https://ru.wikipedia.org/wiki/ARM_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)). Дизайн ядра ARM имеет множество настраиваемых опций, и ST выбирает индивидуальную конфигурацию для каждого микроконтроллера, при этом добавляя свои собственные периферийные устройства к ядру микроконтроллера перед преобразованием дизайна в [полупроводниковую пластину](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%83%D0%BF%D1%80%D0%BE%D0%B2%D0%BE%D0%B4%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%81%D1%82%D0%B8%D0%BD%D0%B0). В следующей таблице представлены основные серии микроконтроллеров семейства STM32.

* 1. **Характеристики микроконтроллера STM32F051R8Tx**
* Ядро: ARM 32-bit Cortex™-M0 CPU, частота до 48 МГц
* Память
  + От 16 до 64 Кбайт флеш-памяти
  + 8 Кбайт SRAM-памяти с проверкой чётности
* Блок вычисления CRC
* Reset and power management
  + Диапазон напряжения: от 2.0В до 3.6В
  + Сброс при включении/выключении питания (POR/PDR)
  + Программируемый детектор напряжения (PVD)
  + Режимы низкой мощности: Сон, Стоп, Ожидание
* Управление таймерами
  + Кварцевый осциллятор, от 4 до 32 МГц
  + Осциллятор для RTC с калибровкой, 32 кГц
  + Встроенный RC-осциллятор 8 МГц
  + Встроенный RC-осциллятор 40 кГц
* До 55 быстрых операций ввода/вывода
  + Все сопоставимы с внешними векторами прерываний
  + До 36 операций с допуском 5В
* 5-канальный DMA контроллер
* Один 12-бит, 1.0 µ АЦП (до 16 каналов)
  + Диапазон преобразования: от 0 до 3.6В
  + Аналоговый источник питания от 2.4 до 3.6В
* Один 12-бит ЦАП
* Два быстрых маломощных аналоговых компаратора с программируемым вводом/выводом
* До 18 емкостных сенсорных каналов с поддержкой сенсорных, линейных и поворотных датчиков
* До 11 таймеров
  + Один 16-битный таймер с расширенными функциями управления для формирования 6 каналов ШИМ
  + Один 32-битный и один 16-битных таймеров с 4 каналами захвата, необходимых для декодирования команд управления инфракрасного порта или управления ЦАП
  + Один 16-битный таймер с 1 IC/OC
  + Независимый и системный сторожевой таймеры
  + Системный таймер SysTick: 24-битный счётчик
  + Один 16-битный обычный таймер для запуска ЦАП
* Часы реального времени (RTC) с календарем, будильником и периодическим выходом из режима останова/ожидания
* Коммуникационные интерфейсы:
  + Два интерфейса I2C с поддержкой режима Fast Mode Plus (скорость передачи данных 1 Мбит/с) и током нагрузки 20 мА, один из которых поддерживает режим шины SMBus/PMBus и выход из режима сна по сигналу пробуждения
  + До 2 портов USART с поддержкой синхронного режима ведущего SPI и управление модемом. Три из них поддерживают интерфейс ISO7816, LIN и IrDA с автоматическим определением скорости передачи данных и функцией выхода из режима сна по сигналу пробуждения
  + Два интерфейса SPI со скоростью передачи данных 18 Мбит/с и программируемой длиной фрейма от 4 до 16 бит, а также мультиплексированный интерфейс I2S
* Функция пробуждения при вставке разъёма в гнездо HDMI
* Последовательный проводной интерфейс отладки (SWD)
* Уникальный 96-битный идентификатор устройства
  1. **Области применения**

Область применения микроконтроллеров безгранична. Их используют в любых электронных устройствах для осуществления контроля. Кроме того, они находятся во всех бытовых приборах – микроволновках, электрочайниках, утюгах, стиральных машинах — микроконтроллер можно запрограммировать под любую функцию.

# **Проектная часть**

## **Описание инструментов разработки программы**

**Среда разработки программного обеспечения**

IAR Embedded Workbench **-** Многофункциональная среда разработки приложений на языках C, C++ и ассемблере для целого ряда микроконтроллеров от различных производителей.

Основные преимущества пакета – дружественный пользовательский интерфейс и непревзойденная оптимизация генерируемого кода. Кроме этого реализована поддержка различных операционных систем реального времени и JTAG -адаптеров сторонних компаний.

В настоящее время IAR Embedded Workbench поддерживает работу с 8-, 16-, 32-разрядными микроконтроллерами от Atmel, ARM, NEC, Infineon, Analog Devices, Cypress, Microchip Technologies, Micronas, Dallas Semiconductor/Maxim, Ember, Luminary, NXP, OKI, Samsung, National Semiconductor, Texas Instruments, STMicroelectronics, Freescale, TI/Chipcon, Silicon Labs и Renesas. Для каждой платформы существует своя среда разработки, в частности ARM микроконтроллерам соответствует версия пакета IAR Embedded Workbench for ARM.

Программная среда включает в себя:

* 1. C/C++ компилятор – один из самых эффективных в своем роде. В нем также присутствует полная поддержка ANSI C.
* 2. Транслятор ассемблера, включающий в себя макроассемблер для программ реального времени и препроцессор для C/C++компилятора.
* 3. Компоновщик, поддерживающий более тридцати различных выходных форматов для совместного использования с внутрисхемными эмуляторами.
* 4. Текстовый редактор, настроенный на синтаксис языка Си и имеющий удобный пользовательский интерфейс, автоматическое выделение ошибок программного кода, настраиваемую инструментальную панель, подсветку блоков, а также удобную навигацию по именам подпрограмм, макросов и переменных.
* 5. Симулятор и отладчик в кодах Си и ассемблера. Отладчик позволяет просматривать области EEPROM, DATA, CODE, а также регистры ввода/вывода, устанавливать точки останова и аппаратные флаги, обрабатывать прерывания с предсказанием. Кроме этого предусмотрен контроль стека и любых локальных переменных, режим пошагового выполнения программы. Тип отладчика и его настройки устанавливаются в свойствах проекта. Если отладчик отсутствует, то на помощь приходит симулятор, который, однако, не имеет возможности эмулировать работу процессора.
* 6. Менеджер проектов, облегчающий контроль и управление рабочими модулями.
* 7. Дополнительные утилиты для работы с оптимизированной CLIB/DLIB библиотекой.

Интегрированная система помощи облегчает написание программ в данной среде. Предусмотрено взаимодействие с утилитой AVR Studio. Помимо прочего в IAR Embedded Workbench существует возможность самостоятельного управления оптимизацией отдельных модулей проекта, что упрощает процесс отладки, а также позволяет ускорить работу программы или сэкономить на памяти.

Язык интерфейса IAR Embedded Workbench – английский.

## **Описание разработанного приложения**

Данное приложение является программой для безопасного заряда Li-Ion аккумулятора, состоит из 2 модулей: Основной код программы и Код системных прерываний.

1. Основной код программы.

Содержит в себе код, выполняемый при подключении микроконтроллера к сети и аккумулятору до возникновения системного прерывания.

1. Код системных прерываний

Содержит в себе код, выполняемый при возникновении системного прерывания (например, окончание итерации таймера)

# **Описание основных функций, используемых в программе**

### **Список файлов**

В состав проекта входят заголовочные файлы, файлы исходного кода, файл ресурсов, текстовые файлы, pro файл и файлы форм. Полный список приведён в табл.1.1

Таблица 1.1.

Список файлов

|  |  |
| --- | --- |
| Наименование файла | Описание |
| startup\_stm32f051x8.s | Файл настройки проекта |
| main.c | Основной код программы |
| stm32f0xx\_it.c | Код системных прерываний |
| stm32f0xx\_hal.c | Файл инициализации драйвера HAL |
| stm32f0xx\_hal\_adc.c | Файл инициализации АЦП |
| stm32f0xx\_hal\_dac.c | Файл инициализации ЦАП |
| stm32f0xx\_hal\_dma.c | Файл инициализации прямого доступа к памяти |
| stm32f0xx\_hal\_exti.c | Файл инициализации системных прерываний |
| stm32f0xx\_hal\_flash.c | Файл инициализации взаимодействия с Flash-памятью |
| stm32f0xx\_hal\_gpio.c | Файл инициализации взаимодействия с пинами ввода/вывода |
| stm32f0xx\_hal\_i2c.c | Файл инициализации шины I2C(используется таймерами) |
| stm32f0xx\_hal\_pwr.c | Файл инициализации регулятора мощности |
| stm32f0xx\_hal\_rcc.c | Файл инициалиации модуля, управляющего тактированием |
| stm32f0xx\_hal\_tim.c | Файл инициализации таймеров |

В таблице 1.2 указаны все функции, в каких файлах они находятся и их описание.

Таблица 1.2

Список функций

|  |  |  |
| --- | --- | --- |
| Наименование функции | Файл | Описание функции |
| int main(void) | main.c | Функция выполнения программы |
| static HAL\_StatusTypeDef ADC\_Enable(ADC\_HandleTypeDef\* hadc); | stm32f0xx\_hal\_adc.c | Функция запуска АЦП |
| static HAL\_StatusTypeDef ADC\_Disable(ADC\_HandleTypeDef\* hadc); | stm32f0xx\_hal\_adc.c | Функция выключения АЦП |
| weak HAL\_StatusTypeDef HAL\_DAC\_Start\_DMA(DAC\_HandleTypeDef\* hdac, uint32\_t Channel, uint32\_t\* pData, uint32\_t Length, uint32\_t Alignment) | stm32f0xx\_hal\_dac.c | Функция запуска ЦАП |
| HAL\_StatusTypeDef HAL\_DAC\_Stop\_DMA(DAC\_HandleTypeDef\* hdac, uint32\_t Channel) | stm32f0xx\_hal\_dac.c | Функция выключения ЦАП |
| void SystemClock\_Config(void) | main.c | Функция конфигурации часов реального времени |
| static void MX\_ADC\_Init(void) | main.c | Функция инициализации АЦП |
| static void MX\_DAC1\_Init(void) | main.c | Функция инициализации ЦАП |
| static void MX\_TIM6\_Init(void) | main.c | Функция инициализации таймера |
| static void MX\_GPIO\_Init(void) | main.c | Функция инициализации взаимодействия с каналами ввода/вывода |
| void Error\_Handler(void) | main.c | Функцция обработчика ошибок |
| void assert\_failed(char \*file, uint32\_t line) | main.c | Функция отчёта об ошибке выполнения |
| void SysTick\_Handler(void) | stm32f0xx\_hal\_tim.c | Функция обработки прерывания системного таймера |
| void ADC1\_COMP\_IRQHandler(void) | stm32f0xx\_hal\_tim.c | Функция обработки прерывания АЦП |
| void TIM6\_DAC\_IRQHandler(void) | stm32f0xx\_hal\_tim.c | Функция обработки прерывания таймера |

Для примера более подробно рассмотрим некоторые функции.

* + 1. Функция тела программы (файл main.c)

Основная функция программы приведена на Листинге 1.1. Комментарии конфигуратора удалены/изменены с целью повышения читабельности.

/\* ПОЛЬЗОВАТЕЛЬСКИЕ ПРИВАТНЫЕ ПЕРЕМЕННЫЕ \*/

unsigned char timSecPassed = 0; //флаг конца итерации таймера

unsigned int timSecCount = 0; //счётчик итераций таймера

unsigned int dacSignal = 0; //переменная значения ЦАП

unsigned char adcCurrentChannel = 0; //текущий канал АЦП

unsigned int adcChannelI = 0; //канал АЦП, хранящий значение тока

unsigned int adcChannelU = 0; //канал АЦП, хранящий значение //напряжения

unsigned char adcEndOfSeq = 0; //флаг конца опроса АЦП входных //каналов

unsigned char phase = 0; //флаг текущего этапа заряда

/\* КОНЕЦ ПОЛЬЗОВАТЕЛЬСКИХ ПРИВАТНЫХ ПЕРЕМЕННЫХ \*/

**Листинг 1.1. Функция main()**

int main(void)

{

/\* Инициализация всех сконфигурированных перифирийных устройств \*/

MX\_GPIO\_Init(); //инициализация входных/выходных каналов

MX\_ADC\_Init(); //инициализация АЦП

MX\_DAC1\_Init(); //инициализация ЦАП

MX\_TIM6\_Init(); //инициализация таймера

/\* НАЧАЛО ПОЛЬЗОВАТЕЛЬСКОГО КОДА 2 \*/

TIM6->CR1 = TIM6->CR1 | 0x01; //включение счётчика таймера

TIM6->DIER = TIM6->DIER | 0x01; //включение прерываний таймера

DAC->CR |= DAC\_CR\_EN1; //включение 1-го канала ЦАП

DAC->CR |= DAC\_CR\_TSEL1; //установка вызывающего события ЦАП

if ((ADC1->ISR & ADC\_ISR\_ADRDY) == 0)//проверка готовности АЦП

ADC1->CR |= ADC\_CR\_ADEN; //включение АЦП

/\* КОНЕЦ ПОЛЬЗОВАТЕЛЬСКОГО КОДА 2 \*/

/\* Бесконечный цикл \*/

/\* НАЧАЛО ПОЛЬЗОВАТЕЛЬСКОГО КОДА WHILE \*/

GPIOC->BSRR = (1<<12)<<16; //установка

timSecPassed = 0; //инициализация флага конца итерации

timSecCount = 0; //инициализация счётчика итераций

dacSignal = 0; //инициализация переменной значения ЦАП

phase = 0; //инициализация переменной этапа заряда

ADC1->IER |= (1<<2)|(1<<3); //включение прерываний АЦП

ADC1->CR |= (1<<2); //опрос АЦП входных каналов

DAC->DHR12R1 = 0; //очищение регистра данных ЦАП

DAC->SWTRIGR = DAC\_SWTRIGR\_SWTRIG1; //включение программных

//триггеров ЦАП

while (1)

{

if (timSecPassed >0)

{

if (timSecCount >= 10)

{

ADC1->CR = (1<<2); //перезапуск АЦП

timSecCount = 0; //сброс счётчика итераций

}

timSecCount++; //увеличение счётчика итераций

timSecPassed = 0; //сброс флага прошедшей итерации

}

if (adcEndOfSeq) //если АЦП опросил входные каналы

{

if (phase == 0) //капельный заряд

{

if ((adcChannelI \* 2800 / 3830) <= 300)//если ток < 3А,

{

if ((adcChannelU \* 15818 / 1387) <= 3700)// если

//напряжение < 3.7В

{

if (dacSignal < 4000) //и сигнал ЦАП < 400

dacSignal++; //увеличение сигнала ЦАП

}

else

phase = 1; //переход к поддержанию заряда

}

else if ((adcChannelI \* 2800 / 3830) >= 500) //если

//ток > 5А

{

if (dacSignal != 0)

dacSignal--; //уменьшение сигнала ЦАП

}

else if ((adcChannelU \* 15818 / 1387) > 3700) //если

//напряжение > 3.7В

phase = 1; //переход к поддержанию заряда

}

else

{

if ((adcChannelU \* 15818 / 1387) < 4000) //если

//напряжение < 5В,

{

if ((adcChannelI \* 2800 / 3830) < 300) //ток < 3А

{

if (dacSignal < 4000) //и сигнал ЦАП < 4000

dacSignal++; //увеличение сигнала ЦАП

}

else if ((adcChannelI \* 2800 / 3830) >= 500) //если

//ток >= 5A

if (dacSignal != 0)

dacSignal--; //уменьшение сигнала ЦАП

}

else if ((adcChannelU \* 15818 / 1387) >= 4200) //если

//напряжение >= 4.2В

if (dacSignal != 0)

dacSignal--; //уменьшение сигнала ЦАП

}

DAC->DHR12R1 = dacSignal; //запись нужного значения в

//регистр ЦАП

DAC->SWTRIGR = DAC\_SWTRIGR\_SWTRIG1; //включение

//программных триггеров ЦАП

adcEndOfSeq = 0; //сброс флага опроса АЦП входных каналов

}

/\* КОНЕЦ ПОЛЬЗОВАТЕЛЬСКОГО КОДА WHILE \*/

}

}

* + 1. Конфигурация таймера (файл main.c)

Конфигурация таймера TIM6 приведёна на Листинге 1.2. Комментарии конфигуратора удалены/изменены с целью повышения читабельности.

**Листинг 2.2. Конфигурация таймера**

static void MX\_TIM6\_Init(void)

{

TIM\_MasterConfigTypeDef sMasterConfig = {0};

htim6.Instance = TIM6; //указание имени таймера

htim6.Init.Prescaler = 3; //установка предделителя

htim6.Init.CounterMode = TIM\_COUNTERMODE\_UP; //установка

//режима счётчика (отсчёт вверх)

htim6.Init.Period = 11999; //установка периода сброса итерации

htim6.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE; //отключение автоперезагрузки при переполнении

if (HAL\_TIM\_Base\_Init(&htim6) != HAL\_OK) //проверка на ошибки

//при инициализации

{

Error\_Handler(); //вызов обработчика ошибок

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET; //сброс //триггеров таймера-Мастера

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE; //отключение режима Master-Slave

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim6, &sMasterConfig) != HAL\_OK) //проверка на ошибки при //синхронизации

{

Error\_Handler(); //вызов обработчика ошибок

}

}

* + 1. Обработка прерывания таймера (файл stm32f0xx\_it.c)

Обработка прерывания таймера приведена на Листинге 1.3. Комментарии конфигуратора удалены/изменены с целью повышения читабельности.

**Листинг 1.3. Обработка прерывания таймера**

void TIM6\_DAC\_IRQHandler(void)

{

/\* НАЧАЛО ПОЛЬЗОВАТЕЛЬСКОГО КОДА TIM6\_DAC\_IRQn 0 \*/

timSecPassed = 1; //установка флага пройденной итерации

/\* КОНЕЦ ПОЛЬЗОВАТЕЛЬСКОГО КОДА TIM6\_DAC\_IRQn 0 \*/

HAL\_TIM\_IRQHandler(&htim6);

HAL\_DAC\_IRQHandler(&hdac1);

}

* + 1. Обработка прерывания АЦП (файл stm32f0xx\_it.c)

Обработка прерывания АЦП приведёна на Листинге 1.4. Комментарии конфигуратора удалены/изменены с целью повышения читабельности.

**Листинг 1.4. Обработка прерывания АЦП**

void ADC1\_COMP\_IRQHandler(void)

{

/\* НАЧАЛО ПОЛЬЗОВАТЕЛЬСКОГО КОДА ADC1\_COMP\_IRQn 0 \*/

unsigned int adcISRvalue; //переменная значения регистра ISR

adcISRvalue = ADC1->ISR; //считывание значения регистра ISR

if ((adcISRvalue & ADC\_ISR\_EOC) != 0) //если значение ISR //получено и АЦП завершил преобразование

{

if (adcCurrentChannel == 0) //если текущий канал АЦП - 0

adcChannelI = ADC1->DR; //запись в соответствующую //переменную значения тока

else if (adcCurrentChannel == 1)//если текущий канал АЦП - 1

adcChannelU = ADC1->DR; //запись в соответствующую //переменную значения напряжения

ADC1->ISR = ADC\_ISR\_EOC; //запись в регистр завершения //преобразования

adcCurrentChannel++; //переход к следующему каналу АЦП

}

adcISRvalue = ADC1->ISR; //считывание значения регистра ISR

if ((adcISRvalue & ADC\_ISR\_EOSEQ) != 0) //если значение ISR //получено и АЦП опросил входные каналы

{

adcEndOfSeq = 1;//установка флага опроса АЦП входных каналов

adcCurrentChannel = 0; //сброс текущего канала АЦП

ADC1->ISR = ADC\_ISR\_EOSEQ; //запись в регистр завершения //опроса входных каналов

}

/\* КОНЕЦ ПОЛЬЗОВАТЕЛЬСКОГО КОДА ADC1\_COMP\_IRQn 0 \*/

HAL\_ADC\_IRQHandler(&hadc);

}

Заключение.

В ходе работы над курсовым проектом, была разработана программа для безопасного и эффективного заряда Li-Ion аккумулятора. были изучены принципы работы с IDE EWARM, которая позволяет программировать микроконтроллеры с архитектурой ARM, также были изучены принципы работы с микроконтроллерами и их регистрами. Так как данная IDE использует язык C, знания в его объектном направлении также были усовершенствованы.

Возможные области применения программы: заряд Li-Ion аккумуляторов.

Были рассмотрены:

1. Инструменты, использовавшиеся при разработке проекта.

2. Разработка программы и её функции.

Список источников.

1. RM0091 Reference manual для микроконтроллеров STM32F0x1/STM32F0x2/STM32F0x8 advanced ARM®-based 32-bit MCUs

<https://www.st.com/content/ccc/resource/technical/document/reference_manual/c2/f8/8a/f2/18/e6/43/96/DM00031936.pdf/files/DM00031936.pdf/jcr:content/translations/en.DM00031936.pdf>

1. Datasheet STM32F051x4 STM32F051x6 STM32F051x8:

<https://pdf1.alldatasheet.com/datasheet-pdf/view/556769/STMICROELECTRONICS/STM32F051R8.html>

1. Википедия. Информация о Li-Ion аккумуляторах:

<https://ru.wikipedia.org/wiki/Литий-ионный_аккумулятор>

1. Сайт EasyElectronics. Описание IAR Embedded Workbench

<http://easyelectronics.ru/arm-uchebnyj-kurs-iar-ewarm-sozdanie-proekta.html>

1. Википедия. Информация об архитектуре ARM

<https://ru.wikipedia.org/wiki/ARM_(архитектура)>

1. Википедия. Информация о семействе микроконтроллеров STM32

<https://ru.wikipedia.org/wiki/STM32>