МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

УЧЕБНЫЙ СИМУЛЯТОР РАБОТЫ PIC МИКРОКОНТРОЛЛЕРА

Студент 30.05.2025 Савицкий Макар Алексеевич
(подпись, дата) (ФИО полностью)

Руководитель 30.05.2025 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 30.05.2025 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 2 июня 2025 г. № 6)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«6»___Мая_____2025__г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент __Савицкий Макар Алексеевич____гр.___21ВВП2_____факультета_ФВТ___

_____

2.Тема работы_ Учебный симулятор работы PIC микроконтроллера._____

Тема утверждена приказом ПГУ №_601/о_____от "_6__"___Мая___2025___г.

3. Руководитель бакалаврской работы___Бычков_А.С._____

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать приложение симулятор, имитирующий работу PIC микроконтроллера на выбранном языке программирования. Программа должна обладать интуитивно понятным интерфейсом и в достаточной для обучения степени демонстрировать работу PIC_микроконтроллра_____
_____
_____
_____
_____

5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметных областей: проанализировать актуальность разработки, изучить существующие симуляторы микроконтроллеров, выделить их достоинства и недостатки. Выбрать язык программирования и микроконтроллер, симуляция которого будет реализована. Выявить необходимые средства взаимодействия с приложением и реализовать их. Описать архитектуру проекта, настроить инструменты разработки, разработать программу, описать реализованные функции, проанализировать результаты, составить рекомендации к дальнейшей разработке._____
_____

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	22	6.05.2025 – 8.05.2025	
2	Анализ требований технического задания	32	9.05.2025 – 12.05.2025	
3	Разработка классов приложения	20	13.05.2025 – 15.05.2025	
4	Реализация интерфейса программы	26	16.05.2025 – 17.05.2025	
5	Разработка основной части программы	90	18.05.2025 – 25.05.2025	
6	Получение и анализ результатов	20	26.05.2025 – 27.05.2025	
7	Оформление пояснительной записки	20	28.05.2025 – 30.05.2025	

Дата выдачи задания "6"___Мая____2025___г.

Руководитель бакалаврской работы _____ 30.05.2025 Бычков А.С.
(подпись, дата)

Задание к исполнению принял студент _____ 30.05.2025 Савицкий М.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 30.05.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 186 листов, 63 рисунка, 1 таблицу, 7 источников, 3 приложения.

МИКРОКОНТРОЛЛЕР, КОМПИЛЯТОР, ОБРАБОТЧИК КОМАНД, ПАМЯТЬ КОМАНД, ПЕРИФЕРИЯ, РЕГИСТРЫ, СИМУЛЯТОР, C#, WINDOWS FORMS

Цель работы – разработать учебное приложение, имитирующее работу PIC микроконтроллера.

Самым простым путем достижения поставленной задачи будет создание монолитного приложения, имитирующего поведение PIC микроконтроллера. Приложение будет разработано на языке программирования C# с использованием удобной технологии Windows Forms. Приложение должно предоставлять возможность написания программы для PIC микроконтроллера, возможность ее проверки, компиляции в машинный код, и последующей отладки команд. Все области памяти должны быть доступны для просмотра пользователем. Также необходимо добавить различные способы обработки команд, для удобной работы с симулятором.

В итоге было разработано приложение, которое, которое демонстрирует работу микроконтроллера в достаточной степени, чтобы на нем изучать работы PIC микроконтроллера. В ходе разработки, была использованы технологии многопоточного программирования и Windows Forms. Реализованный на данный момент функционал приложения протестирован и готов к эксплуатации.

Содержание

Перечень сокращений и обозначений 7

Введение 8

1 Анализ предметной области 10

1.1 Анализ решения поставленной задачи 11

1.2 Анализ актуальности разработки 12

1.3 Анализ аналогичных решений 13

1.4 Постановка задачи 19

1.5 Описание архитектуры проекта 21

1.6 Выбор инструментов разработки 23

1.6.1 Выбор языка программирования 23

1.6.2 Выбор микроконтроллера 25

2 Реализация 27

2.1 Установка и настройка инструментов разработки 27

2.2 Описание структуры программы 30

2.3 Разработка программы 33

2.3.1 Разработка компилятора 33

2.3.2 Разработка отладчика 35

2.3.3 Разработка периферии 38

2.3.4 Разработка обработчика команд 38

2.3.5 Разработка класса микроконтроллера 40

2.4 Описание доступных функций 40

2.1.1. Описание работы компилятора 41

2.1.2 Описание работы отладчика 43

2.1.3 Описание работы периферии 49
3 Анализ результатов 53
3.1 Результаты разработки приложения симулятора 53
3.2 Рекомендации к дальнейшей разработке 55
4 Безопасность жизнедеятельности 57
4.1 Правила безопасности при использовании ПК 57
Заключение 58
Список использованных источников 59
Приложение А – UML-диаграммы 60
Приложение Б – Листинг программы 65
Приложение В – Презентация 186

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

СК (счетчик команд) - регистр процессора, который указывает, какую команду нужно выполнять следующей.

JavaFX - платформа на основе Java для создания приложений с насыщенным графическим интерфейсом.

JIT-компиляции - тип компиляции, который происходит во время выполнения программы.

RAM (Random Access Memory) - тип компьютерной памяти, который позволяет быстро получать доступ к любым данным, хранящимся в нем.

RC-генератор - генераторы, в которых в качестве частотно задающего элемента используются сопротивления и конденсаторы.

RISC-архитектура - архитектурный подход к проектированию процессоров, в которой быстродействие увеличивается за счёт такого кодирования инструкций, чтобы их декодирование было более простым, а время выполнения — меньшим.

Program Memory - область памяти, где хранятся и выполняются инструкции программы.

PyQt5 - набор расширений графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.

UML (Unified Modeling Language) – унифицированный язык моделирования.

WDT (Watchdog Timer) - система контроля зависания устройства с последующим его перезапуском.

Windows Forms - визуальная поверхность, на которой выводится информация для пользователя.

Wreg - рабочий регистр в микроконтроллерах PIC.

Введение

В нашем мире уже почти все аспекты окружающей жизни переданы на управлении различной электронике: телекоммуникация, бытовая техника, энергосистемы, промышленная автоматика и т.д. Абсолютно каждую сферу жизни люди пытаются автоматизировать для оптимизации ежедневной рутины облегчения своей жизни. Ведь зачем выделять свое время и силы для управления различными однотипными задачами, если это можно передать под управление автономному устройству, контроллеру.

Для выбора, подходящего под условия задачи устройства автоматизации в основном, используют два подхода: микрокомпьютер и микроконтроллер. У каждого из подходов есть свои преимущества и недостатки. Так, например, из преимуществ у микрокомпьютера имеется более мощный микропроцессор, на котором можно запустить полноценную операционную систему, более простая разработка программного обеспечения и наличие стандартных интерфейсов подключения. Но в отличии от микроконтроллеров, микрокомпьютеры имеют большую уязвимость для вирусов, меньшую надежность, высокое энергопотребление и большую стоимость. И если устройство выйдет из строя, понадобиться гораздо больше времени для возврата работоспособности системы. Микроконтроллеры же несмотря на преимущества более дешевого и простого устройства имеют большую сложность программирования и использования существующих наработок.

В данной выпускной работе будет рассмотрено следующее приложение: “Учебный симулятор работы PIC микроконтроллера”, который поможет с пониманием того, как устроены и как работают PIC микроконтроллеры. Данный симулятор позволяет отладить программу на различные ошибки, дает возможность просмотреть результат компиляции программы в машинном коде, а также предоставляет возможность взаимодействия с различной периферией микроконтроллера, что дает возможность повысить качество обучения программирования данных микроконтроллеров.

Цели выпускной квалификационной работы:

– освоение современных средств программной разработки;

– исследование существующих аналогичных приложений симуляторов;

– исследование преимуществ симуляторов над реальными устройствами в обучающих целях;

– определение требований к функциональности и интерфейсу разрабатываемого приложения;

– определение требований к внутренней работе приложения, имитирующей поведение микроконтроллера;

– определение требований к функционалу возможностям отладки программного кода;

– разработка учебного симулятора работы PIC микроконтроллера;

– проведение тестирования разработанного приложения в его функциональности и надежности;

– анализ полученных результатов, полученных в ходе проведения тестирования микроконтроллеров в обучающих целях;

– сделать выводы о преимуществах и потенциале использования симуляторов микроконтроллеров в обучающих целях;

– предложить рекомендации по дальнейшему развитию и совершенствованию приложения симулятора.

1 Анализ предметной области

Для разработки качественного приложения компиляции кода и симулятора, имитирующего поведение и работу PIC микроконтроллера необходимо проанализировать все аспекты работы данного приложения. Необходимо понимать, какой набор возможностей в программировании имеется, какой набор команд существует, как выполнена работа с памятью и как работает обработчик команд микроконтроллера. В рамках данного раздела будет рассмотрен ряд технических моментов, из которых будет понятно, как корректно реализовать “Учебный симулятор работы PIC микроконтроллера”.

Первым этапом при разработке данного приложения будет анализ существующих подходов разработки и тестирования программ при работе с микроконтроллерами, при котором было выявлено несколько основных способов разработки программы.

Каждому, кто начинает работать или уже работает в программировании микроконтроллеров, необходимо написать и отладить программу на специальном ПО, так чтобы готовый код программы не выдавал ошибок. Затем зачастую для тестирования программы требуется физический микроконтроллер, в память которого загружается программа. После чего микроконтроллер правильно включается и выполняет загруженную программу. Но если программу можно найти и спокойно скачать, а в случае сбоя запросто переустановить, то при поломке микроконтроллера в ходе обучения придется его полностью заменить на новый. Также при данном подходе в любом случае для обучения нужно будет обзавестись хотя бы одним PIC микроконтроллером на каждого обучающегося специалиста. Это все к дополнительным расходам на обучение.

Однако есть и другой способ разработки и тестирования программы, который не требует наличия физического PIC микроконтроллера. Это возможность симуляции работы программы напрямую в приложении компилятора. Также в отличие от первого подхода разработки программы, здесь

можно в любой момент исполнения программы просмотреть состояние различных регистров, построчно исполнять код для нахождения неисправностей в работе программы и возможность полноценно взаимодействовать с периферией устройства, имитируя ее работу. Также при работе на симуляторе нет необходимости переживать что программа может вывести контроллер из строя. Все аспекты, описанные выше, в совокупности дают большие преимущества при обучении в программировании микроконтроллера именно на симуляторе.

В ходе анализа темы были обозначены проблемы, связанные с текущими системами разработки программ для работы микроконтроллеров. К ним относится сложность просмотра регистров состояний, ограниченность эмуляции периферии микроконтроллера, и многое другое. Также были определены требования для работы приложения “Учебный симулятор PIC микроконтроллера”. Они включают в себя: удобство использования, скорость работы, доступность и надежность.

В результате анализа программирования микроконтроллеров стало понятно, что приложения “Учебный симулятор PIC микроконтроллера” может повысить степень обучаемости программирования PIC микроконтроллеров и степень понимания работы микроконтроллера.

1.1 Анализ решения поставленной задачи

Симулятор — это программа, которая необходима для отображения работы какого-либо процесса подражая реальному устройству, с которым можно взаимодействовать через возможности интерфейса. Симуляторы микроконтроллеров позволяют наглядно и безопасно продемонстрировать его работу на практике: отобразить состояние, взаимодействовать через эмуляторы периферии. Это отлично подходит для как для обучения начинающих специалистов в данной сфере, так и для инженеров, которым необходимо отладить продукт, не имея физического микроконтроллера под рукой. Рассмотрим некоторые из преимуществ симуляторов микроконтроллеров:

– возможность обучения, без опасения поломки устройства. Безопасность для различных экспериментов с устройством;

– ускорение процесса написания и отладки программы. Для тестирования отсутствует необходимость в наличии физических контроллеров;

– расширенные возможности отладки. Возможность просмотреть полное состояние устройства;

– быстрота отладки. Работу программируемого устройства можно сразу просмотреть в симуляторе;

– простота использования. В симуляторе имеются все необходимые элементы для его работы;

– доступность. Для начала изучения работы устройства необходимо только скачать сам симулятор;

– отсутствие затрат на обучение. Так как отсутствует потребность в использовании физического устройства;

– гибкость. Необходимые параметры для работы симулятора легко можно настроить в самом приложении.

1.2 Анализ актуальности разработки

В мире, где уже огромная часть задач нашей жизни передана под контроль и исполнение микроконтроллерам, разработка различных симуляторов микроконтроллеров для обучения программированию и возможности отладки программ имеет большое значение.

Во-первых, обслуживание симуляторов микроконтроллеров гораздо выгоднее нежели использование реальных устройств, что используется повсеместно.

Во-вторых, при использовании симуляторов вы обучаетесь почти в полноценных рабочих условиях без всяких рисков. И получив необходимый опыт работы с устройством, вы можете начинать работать уже на реальных микроконтроллерах

В-третьих, разобрав в мельчайших деталях все процессы, протекающие в микроконтроллере на симуляторе, у вас будет понимание данного устройства, для проектирования уже своего программируемого устройства.

Также следует отметить, что при поломке устройств необходимо время для покупки нового и для замены старого, однако эта необходимость пропадает при использовании симуляторов. Таким образом, предложенный проект имеет высокую значимость и актуальность для начинающих специалистов, а также для развития и усовершенствования направления разработки на микроконтроллерах в целом.

1.3 Анализ аналогичных решений

Анализ аналогичных решений является неотъемлемой частью разработки любого приложения, так как дает возможность рассмотреть и изучить подобные решения, выявив их преимущества и недостатки, тем самым определив необходимы для реализации в приложении аспекты. В нашем случае я исследовал различные приложения для обучения разработке программного обеспечения для микроконтроллера.

В качестве первого аналога рассмотрим Симулятор МК8051, который встречался в нашем учебном курсе, на котором для изучения программирования и работы микроконтроллера МК8051 был выделен отдельный семестр. Данный симулятор имеет заранее составленную схему подключения периферийных устройств. Отображение приложения «Симулятор МК8051» рассмотрено на рисунке 1.1.

Данное приложение имеет возможность отслеживать состояние управляющих регистров и портов ввода-вывода информации с микроконтроллера. В нем представлена эмуляция различных периферийных устройств, таких как набор переключателей, клавиатура, полоска светодиодов, семисегментный индикатор на 4 разряда, LED дисплей, компаратор, аналого-цифровой

преобразователь, цифро-аналоговый преобразователь и универсальный асинхронный приемопередатчик.

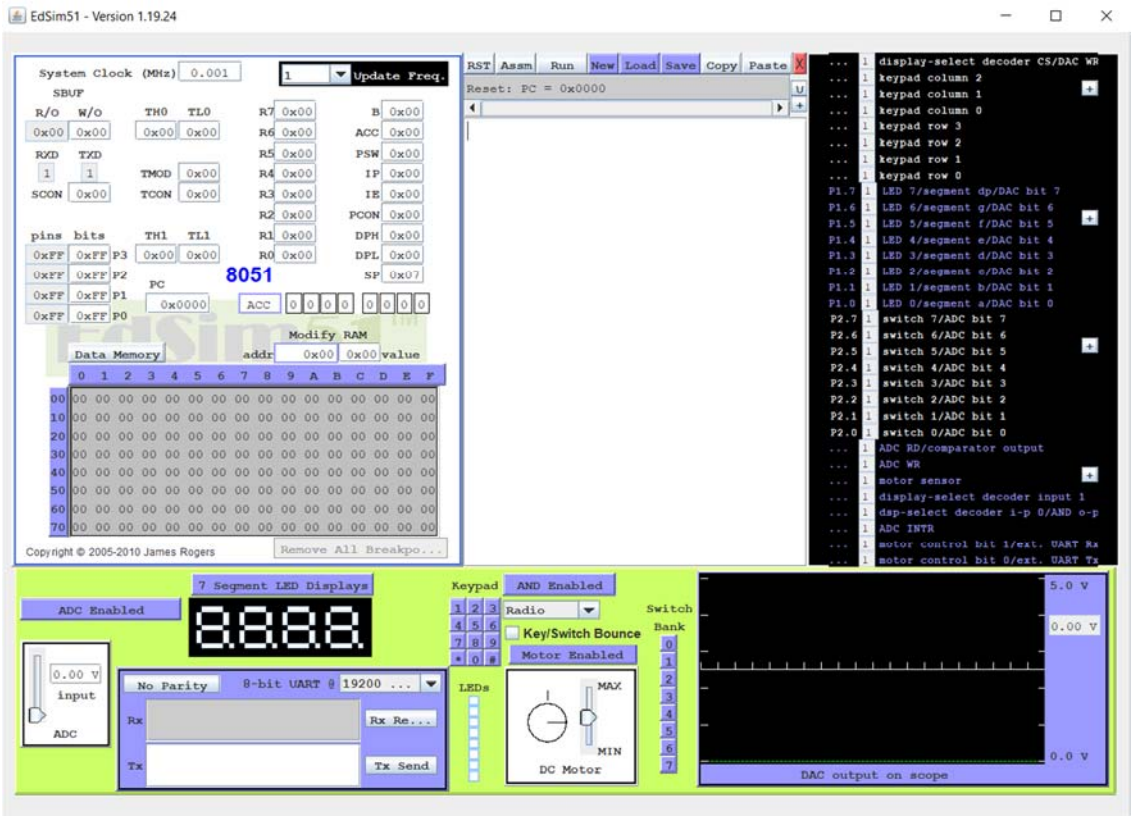


Рисунок 1.1 – Обзор аналога «Симулятор МК8051»

Также имеется возможность просмотреть машинный код программы и состояние оперативной памяти. В приложении реализована возможность просмотреть возможные ошибки при компиляции программы. Но при компиляции программы, код просматривается только до первой найденной ошибки, после чего выводит строку, в которой она допущена и краткое описание ошибки. То есть если в коде имеется множество ошибок, то необходимо каждый раз запускать программу и устранять ошибки по одной. Что представляет сложность для пользователя программы.

Следующим на очереди стало онлайн приложение “WOKWI”, предназначенное для симуляции различных плат Arduino, Отображение приложения «Симулятор МК8051» рассмотрено на рисунке 1.2.

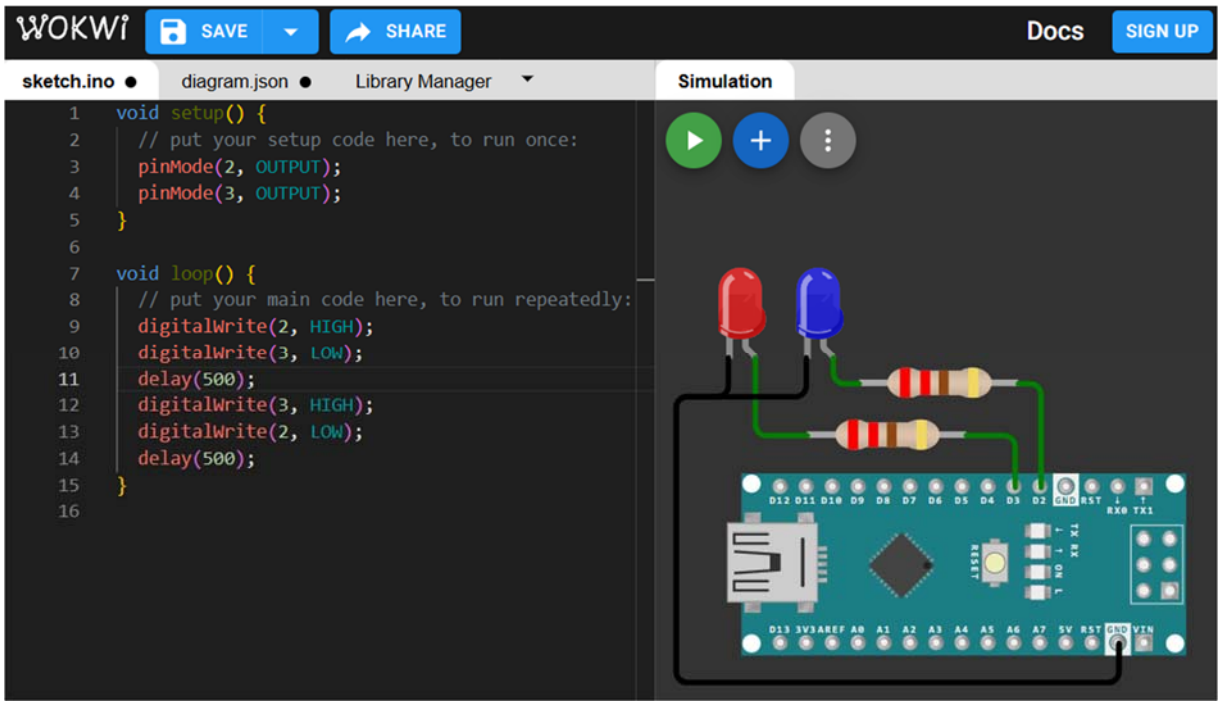


Рисунок 1.2 – Обзор аналога «WOKWI»

Большим достоинством приложение является возможность собрать совершенно любую схему подключения периферии к микроконтроллеру, в данном примере были подключены светодиоды, работающие как мигалка. Но запас различных радиоэлементов довольно большой в имеющейся базе данных. Компилятор довольно удобный и сразу же при написании кода отображает возможные ошибки в нем. Однако данное приложение для своей работоспособности требует подключения к сети интернет, а также отсутствует возможность узнать состояние памяти и регистров микроконтроллера, те нет возможность “заглянуть под капот” микроконтроллера.

В следующем примере рассмотрим совместную работу приложений “OVEN Logic” и “Программа конфигуратор СП200” от отечественного разработчика “ОВЕН”. Данные программы имитируют работу панель оператора и работу контроллера на практике данные устройства соединены интерфейсом RS-485. Отображение приложений рассмотрено на рисунках 1.3 и 1.4.

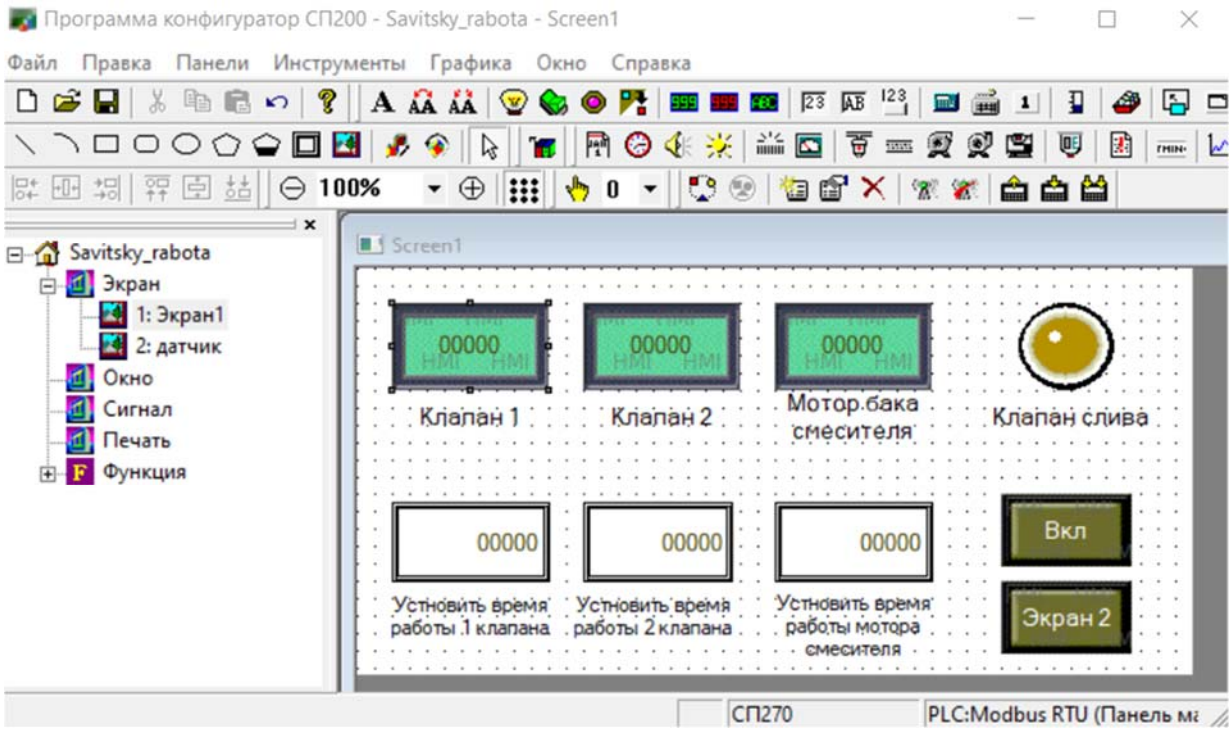


Рисунок 1.3 – Обзор аналога «Программа конфигуратор СП200»

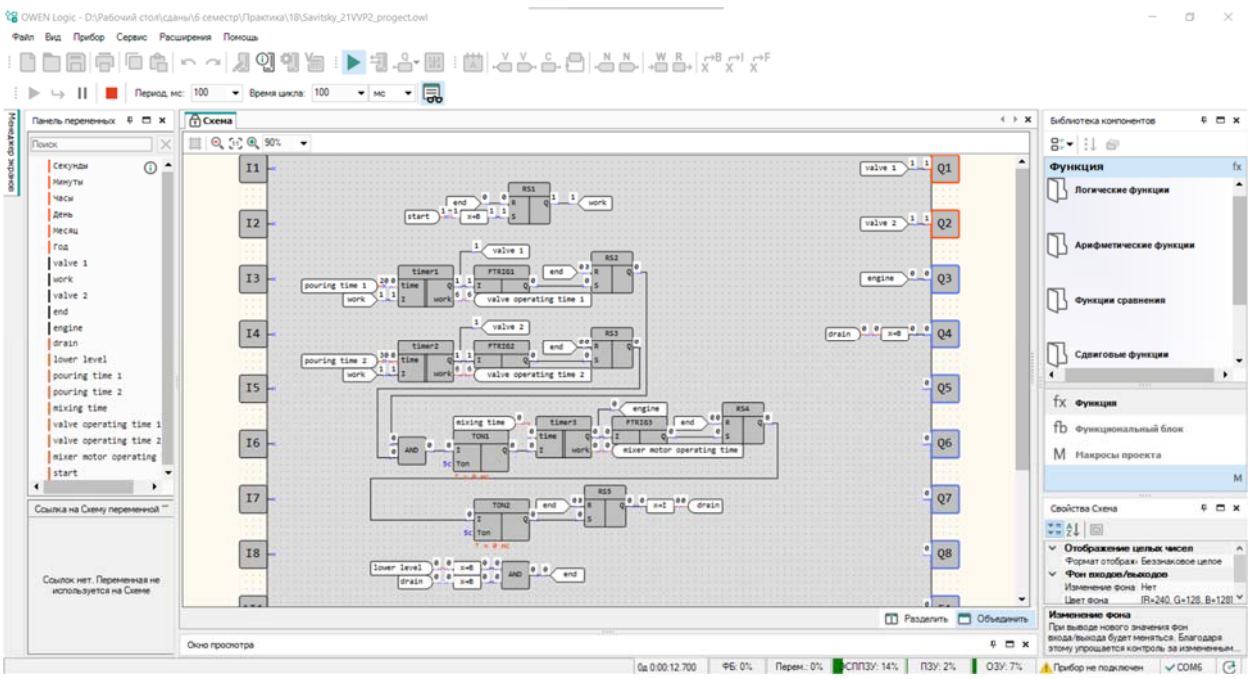


Рисунок 1.4 – Обзор аналога «OVEN Logic»

В данных приложения имеется возможность как составить свою панель оператора, отображающую состояние контроллера, а также взаимодействовать с ним, так и возможность проверить работоспособность программы в режиме

симуляции, где вся программа выглядит в виде блок-схем, в значения переменных видно на текущей схеме. Однако взаимосвязь программ в симуляции отсутствует возможность просмотреть и программы связаны только с помощью указания одинаковых адресов регистров.

Напоследок в качестве примера рассмотрим приложение “MPLAB IDE” от компании “Microchip Technology”, которая разрабатывает и выпускает PIC микроконтроллеры. Отображение приложения «MPLAB IDE» рассмотрено на рисунке 1.5.

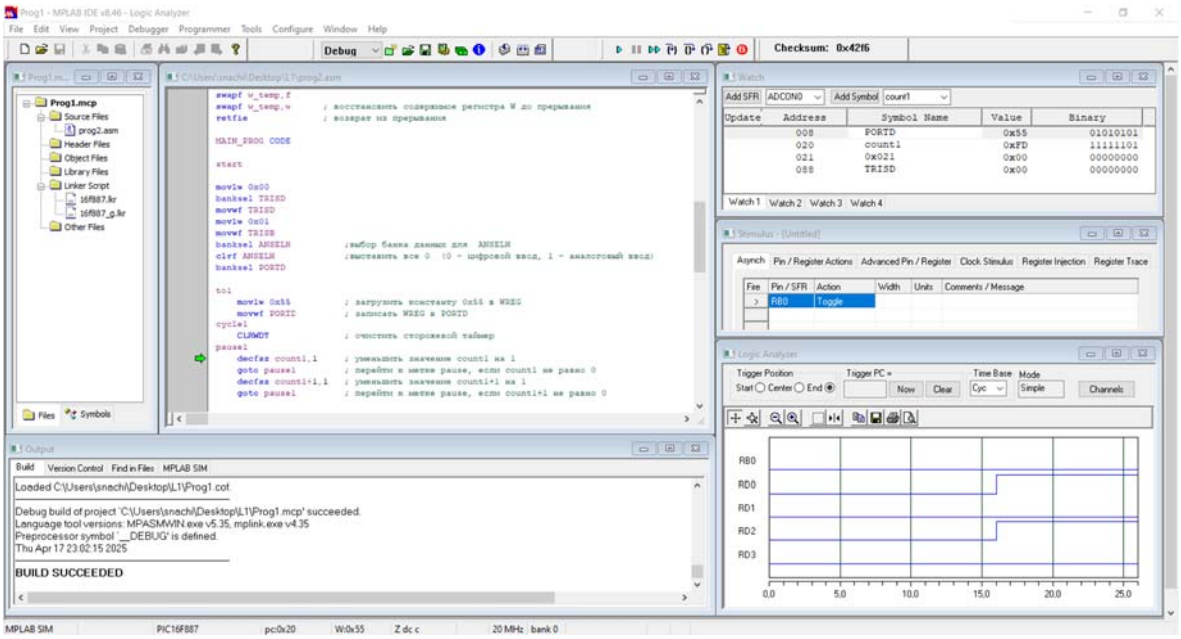


Рисунок 1.5 – Обзор аналога «MPLAB IDE»

В данном приложении реализовано несколько режимов работы, загрузка программного кода на различные платы, а также работа в режиме симуляции. Возможности компилятора позволяют с удобством отладить программу, при запуске которой отмечаются все имеющиеся ошибки в написанном коде. Эмуляция периферии выполнена в виде простых кнопок, то есть имеется возможности подавать только 0 или 1 на контакты, при этом в PIC микроконтроллере имеется возможность считывания аналогового сигнала, что невозможно проверить в режиме симуляции. В данном приложения имеются окна для отслеживания сигналов на контактах в режиме графика на временной шкале,

что довольно удобно для изучения работы микроконтроллера. Так же имеется возможность отслеживать различные служебные регистры и ячейки ОЗУ. Однако можно выделить не совсем интуитивно понятный интерфейс, на практике очень часто наблюдалось, что начинающие пользователи симулятора довольно долго адаптировались к предоставляемым возможностям данной программы.

Рассмотрев все сильные и слабые стороны аналогов, мы можем подвести итог в виде краткого содержание всей информации об этих приложениях, что представлено в таблице 1.

Таблица 1 – Краткое содержание аналоговых решений

Название аналога	Преимущества аналога	Недостатки аналога
Симулятор микропроцессора МК8051	Большой функционал периферии Возможность просмотреть регистры состояния Возможность просмотреть машинный код и состояние оперативной памяти	Неудобная отладка программы
Онлайн приложение “WOKWI”	Удобство работы с компилятором Большая база данных с периферией	Требуется онлайн подключения Отсутствует возможность просмотра состояния памяти
Приложения “Программа конфигуратор СП200” и “ OVEN Logic”	Наглядный формат написания кода Наглядный процесс исполнения кода	Отсутствие возможности продемонстрировать совместную работу Отсутствует возможность просмотра состояния памяти

Название аналога	Преимущества аналога	Недостатки аналога
Приложение “MPLAB IDE”	Удобная отладка программы Просмотр параметров программы на временной шкале Возможность просмотра регистров	Отсутствие полноценной работы с периферией Интуитивно непонятный интерфейс программы

В целом все эти аналогичные решения реализуют в себе некоторые возможности для разработки и отладки кода, однако существующий аналог для PIC микроконтроллеров имеет ряд недостатков, которые будут исправлены в ходе выполнения данной работы.

1.4 Постановка задачи

Необходимо определить инструментарий в работе приложения “Учебный симулятор PIC микроконтроллера”. Приложение должно предоставлять возможность отображать допущенные ошибки в коде программы при компиляции и при работе с периферией, реализовывать возможность просмотра состояния RAM и Program Memory, а также различных предделителей, стека и внутренний памяти. Иметь удобные инструменты отладки такие как: возможность покомандной обработки, возможность обработки команд циклически, установка точек остановки, настройка задержки при выполнении команд, отображение текущего положения обработчика команд, отображение процессорного времени, возможность просмотра памяти контроллера, а также в достаточной степени имитировать работу периферии PIC микроконтроллера.

При первом открытии приложения в доступ пользователю предоставляется компилятор позволять написать предполагаемый код программы и иметь возможность проверить код и указать пользователю на все ошибки допущенные в

нем. Так же при отсутствии ошибок в разрабатываемой программе компилятор составляет машинный код программы, который располагается в памяти микроконтроллера. Подробную диаграмму прецедентов интерфейса компилятора можно изучить на рисунке 1.5.

Рисунок 1.5 – UML диаграмма прецедентов интерфейса компилятора

Кроме того, в состав приложения симулятора будет иметь интерфейс с возможностями для отладки. Пользователю предоставляется возможность обработки команд как пошагово, так и через цикл с возможностью расстановки точек остановки. Так же имеется возможность просматривать внутреннее состояние оперативной памяти контроллера. Подробную диаграмму прецедентов интерфейса обработчика команд можно изучить на рисунке 1.6.

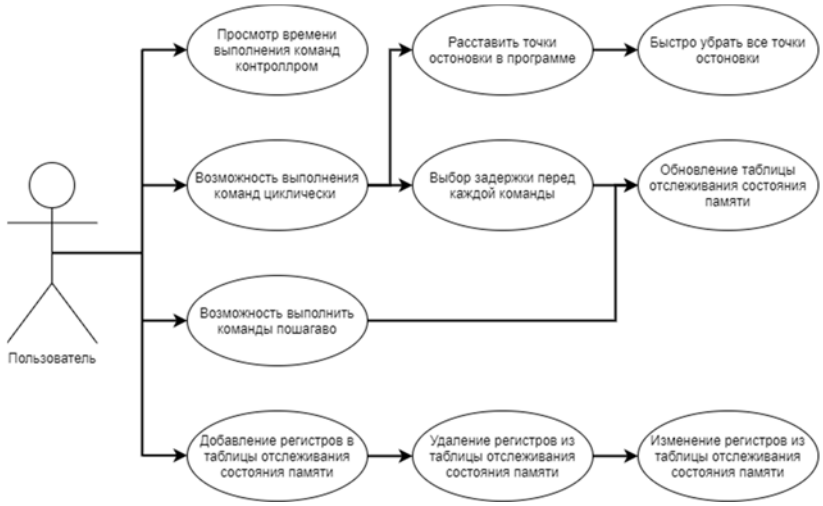


Рисунок 1.6 – UML диаграмма прецедентов интерфейса отладчика

Также, в приложении должен быть реализован интерфейс взаимодействия с периферией. Пользователь должен иметь возможность подачи различный видов сигналов на контроллер, а также наблюдать выводимые с контроллера сигналы. Подробную диаграмму прецедентов интерфейса периферии можно изучить на рисунках 1.7.

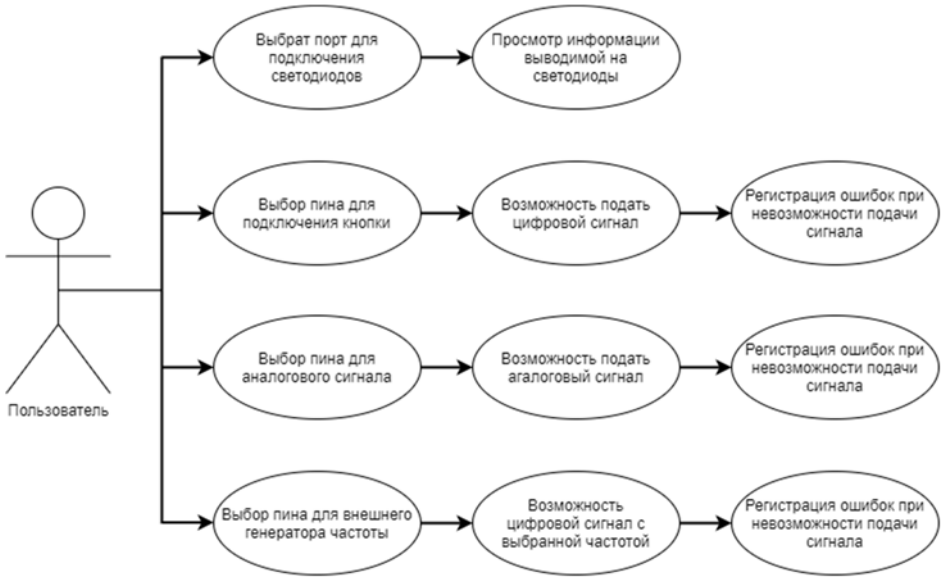


Рисунок 1.7 – UML диаграмма прецедентов интерфейса периферии

В ходе разработки приложения также необходимо придерживаться ее основная цели – это упрощение процесса обучения программированию и пониманию работы PIC микроконтроллера путем предоставления большего контроля над контроллером и его состояния. Симулятор должен быть удобен и прост в использовании, а также обладать высокой надежностью работы.

1.5 Описание архитектуры проекта

После того как мы определили цели разрабатываемого приложения, необходимо определить его вид. Удобнее всего с симулятором будет работать на основе десктопного приложения с использованием монолитной архитектуры.

Монолитная архитектура представляет из себя единый модуль, который может работать автономно, и независимо от других приложений. Благодаря

такому выбору архитектуры пользователю не нужно будет заморачиваться с установкой каких-либо других приложений, а для начала работы необходимо будет просто включить наш разрабатываемый симулятор. Также данная архитектура имеет рад удобных для нас преимуществ: удобная отладка, упрощённое тестирование и высокая производительность.

Приложение будет в себя включать интерфейс, блок компиляции кода, и блок симуляции работы микроконтроллера. Используем архитектуру приложения, изображенную на рисунке 1.8.

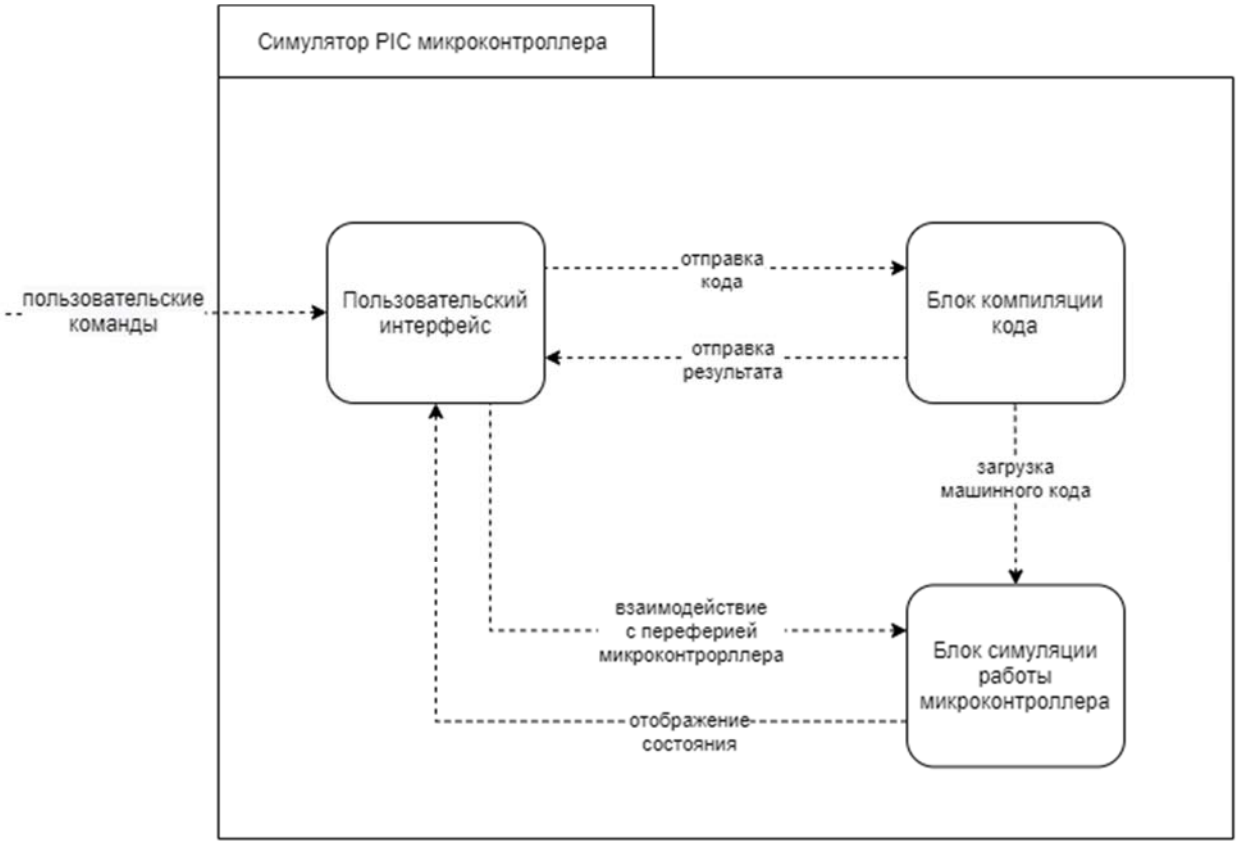


Рисунок 1.8 – Архитектура приложения

В пользовательском интерфейсе будет располагаться редактор кода, в котором пользователь может написать тестируемую программу и отладить ее при нажатии на кнопку компиляции. Так же будут располагаться поля, отображающие состояние внутренних регистров микроконтроллера.

При нажатии на кнопку компиляции написанная программа обрабатывается блоком компиляции кода, происходит проверка кода на наличие ошибок, которые отображаются в пользовательском интерфейсе при их нахождении. Если же ошибок не будет найдено формируется машинный код, который отправляется в блок симуляции работы PIC микроконтроллера.

Исполнение готовой программы на блоке симулятора PIC микроконтроллера можно будет запустить как в виде безостановочного выполнения, так и по точкам остановки. При выполнении программы симулятор будет отображать такие данные процессора как состояния памяти программ и д памяти данных, а также управляющие регистры.

При успешной передаче кода программы в симулятор микроконтроллера на пользовательском интерфейсе станет доступно управление эмулятором периферии микроконтроллера. Через пользовательский интерфейс будет возможно как ввод и вывод цифровых сигналов и контактов микроконтроллера, так и аналоговые для демонстрации работы аналого-цифрового преобразователя.

В пользовательском интерфейсе также будут располагаться инструменты для сохранения разрабатываемого проекта и для его загрузки, что облегчит разработку.

1.6 Выбор инструментов разработки

При разработке симулятора микроконтроллера необходимо с ответственностью подойти к выбору инструментов разработки. Необходимо определиться с выбором языка программирования, на котором будет написано приложение и с реализуемой микросхемой, работу которой будет имитировать приложение.

1.6.1 Выбор языка программирования

При разработке симулятора микроконтроллера необходимо с ответственностью подойти к выбору инструментов разработки. Необходимо определиться с выбором языка программирования, на котором будет написано

приложение и с реализуемой микросхемой, работу которой будет имитировать приложение.

В настоящие время для написания своего десктопного приложения существует огромный выбор языков программирования такие как C++, C#, Python, Java.

Python – это высокоуровневый язык программирования, отличающийся своей эффективностью и простотой использования, за счет простого и интуитивно понятного для разработчиков синтаксиса. Код на данном языке программирования можно запустить на любой платформе, при условии, что на ней установлен интерпретатор, который последовательно исполняются операторы. Если в процессе обработки кода программа-интерпретатор сталкивается с ошибкой, исполнение прекращается. Это помогает быстро найти и исправить ошибку. Из года в год все больше молодые разработчики выбирают Python своим первым языком программирования. При выборе данного языка программирования будет проще найти единомышленников для будущего развития проекта. Однако для разработки графического интерфейса программы необходимо будет подключить фреймворк PyQt5, благодаря которому разработка упроститься в несколько раз.

Java является довольно известным объектно-ориентированным языком программирования. В отличии от ранее представленного Python, у Java статическая типизация, при вводе переменной необходимо указывать ее тип данных, что позволит избежать большой путаницы при написании кода. Так же так как Java компилирует заранее при помощи JIT-компиляции, а Python — при выполнении программы, то он получает выигрыш в скорости и производительности, однако в читаемости написанного кода все же уступает своему конкуренту. При выборе данного языка программирования так же понадобиться фреймворк JavaFX.

Также довольно популярным для разработка является язык C++. Он имеет ряд значительных особенностей, которые выделяют его на фоне остальных языков программирования. Данный язык программирования был разработан на основе

языка C, который поддерживает как современный набор конструкций структурного программирования, так и ряд низкоуровневых черт. Но в отличии от языка C, C++ обеспечивает гибкие и эффективные средства определения новых типов. На C++ в сравнении с Java куда проще оптимизировать работу с памятью и с вычислительными мощностями, но он сложнее в изучении и скорости написания программы. Однако написание графического интерфейса на C++ является довольно трудной задачей, так что данный язык подходит по большей степени для описания внутренней работы приложения.

C# является более молодым языком программирования, его синтаксис напоминает C++ и Java. C# имеет очень богатый синтаксис, который одновременно с этим простой и удобный. Одним из основных достоинств C# также, как и в языке Java является наличие автоматического управления памяти, но C# выигрывает в плане производительности. Еще одним преимуществом языка C# является простейшая технология для создания графических приложений Windows Forms.

Учитывая все преимущества и недостатки представленных языков программирования – был выбран язык C# [1,2] с использованием технологии Windows Forms [3], так же при разработке приложения хорошим решением будет посещаться форумы [4] для решения различных вопросов.

1.6.2 Выбор микроконтроллера

PIC это название серии микроконтроллеров, выпускаемых компанией Microchip Technology. Данные микроконтроллеры имеют RISC-архитектуру. Данная архитектура подразумевает использование отдельных друг от друга наборов команд для работы с памятью и наборов команд для преобразования информации в регистрах процессора. Наиболее рекомендуемой для новичков является 8-битная линейка микроконтроллеров. Все 8-битные PIC микроконтроллеры разделяются на 3 семейства: Baseline, Mid-range и Enhanced Mid-range, у каждого из которых различная архитектура и различный набор команд.

Для создания учебного симулятора выберем микроконтроллер PIC16F887 [5] из семейства Mid-range, который относиться к группе наиболее полных представителей семейства. Он является полностью статическим устройством, что означает, что он работает с ограниченным объемом памяти программ в 8192 14-разрядных слов и памяти данных в 512 8-разрядных байт. При этом в памяти данных 96 байт выделены под управляющие регистры, 48 байт выделены под регистры специальных функций, а оставшиеся 368 байт являются регистрами общего назначения, куда можно сохранять различные переменные необходимые для работы программы. Также можно выделить простую систему прерываний, 8-уровневый 13-разрядный аппаратный стек (при переполнении которого происходит девятая запись заменяет первую), сторожевой таймер WDT с встроенным RC-генератором (который может инициировать процесс сброса микроконтроллера при аномальном поведении системы), программный выбор параметров тактового генератора. В арифметико-логическое устройство PIC микроконтроллера имеется высокоскоростная архитектура команд, вм вмещающая в себя 35 инструкций, различные способы адресации: прямая, косвенная и относительная. В данном PIC микроконтроллере все команды выполняются за один командный цикл (4 такта) кроме инструкций переходов, выполняемых за два цикла.

Все вышеописанные характеристики отлично демонстрируют возможности PIC микроконтроллеров и идеально подходят для изучения архитектуры микроконтроллера и обучения программированию на нем.

2 Реализация

2.1 Установка и настройка инструментов разработки

Перед началом написания программы на C# необходимо установить программу позволяющую работать с данным языком. Нужно перейти на официальный сайт «https://visualstudio.microsoft.com/ru/downloads/» и скачать бесплатную и полнофункциональную среду Visual Studio Community 2022 как показано на рисунке 2.1.

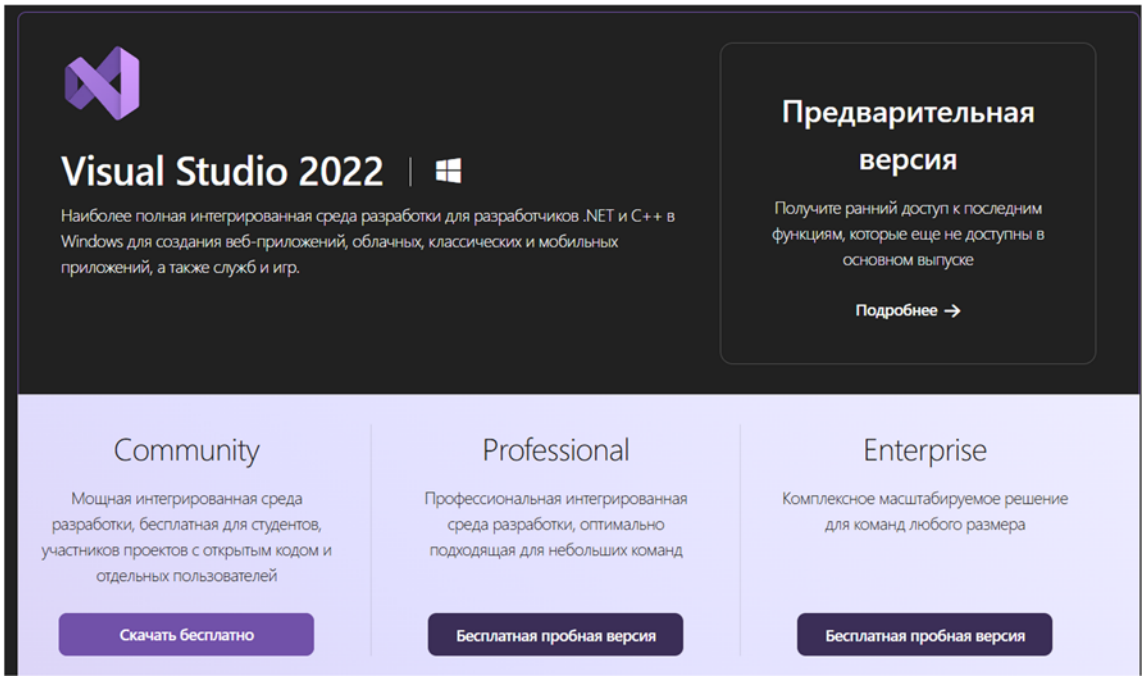


Рисунок 2.1 – Страница загрузки установщика Visual Studio 2022

После завершения загрузки запускаем данный установщик. В открывшемся окне нам необходимо выбрать в качестве компетентна “Разработка классических приложений .NET” как показано на картинке 2.2. Это позволит нам работать с Windows Forms.

Далее необходимо создать проект, в котором будем разрабатывать наше приложение, переходим в установленный ранее Visual Studio 2022 и выбираем пункт “Создание проекта” как показано на рисунке 2.3.

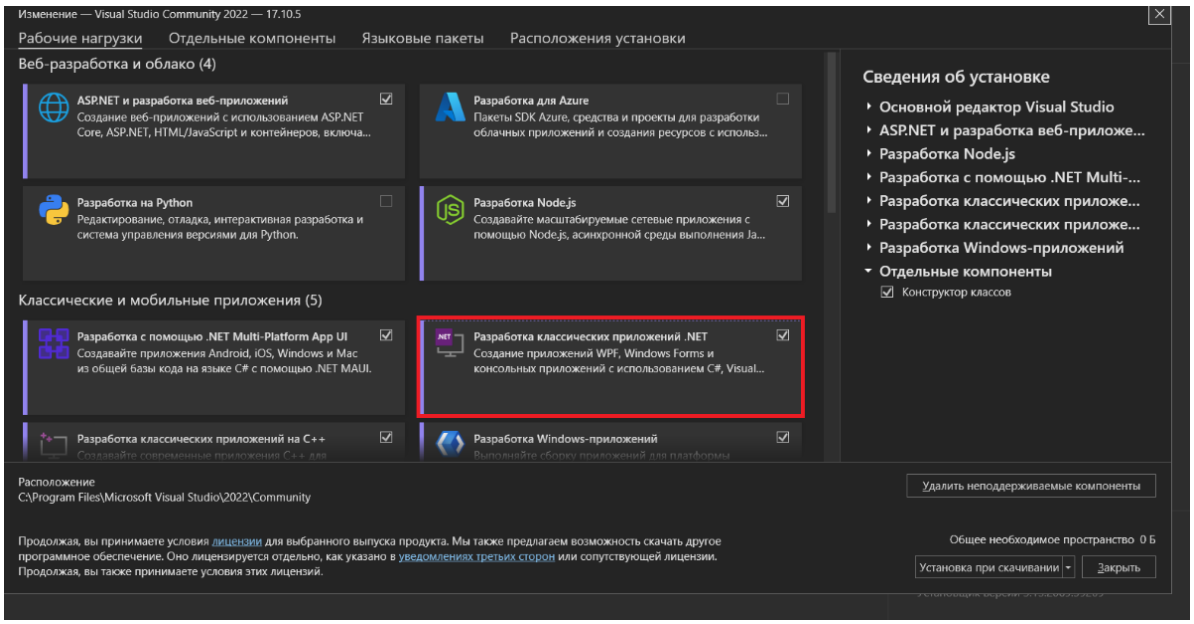


Рисунок 2.2 – Выбор компонентов для установки

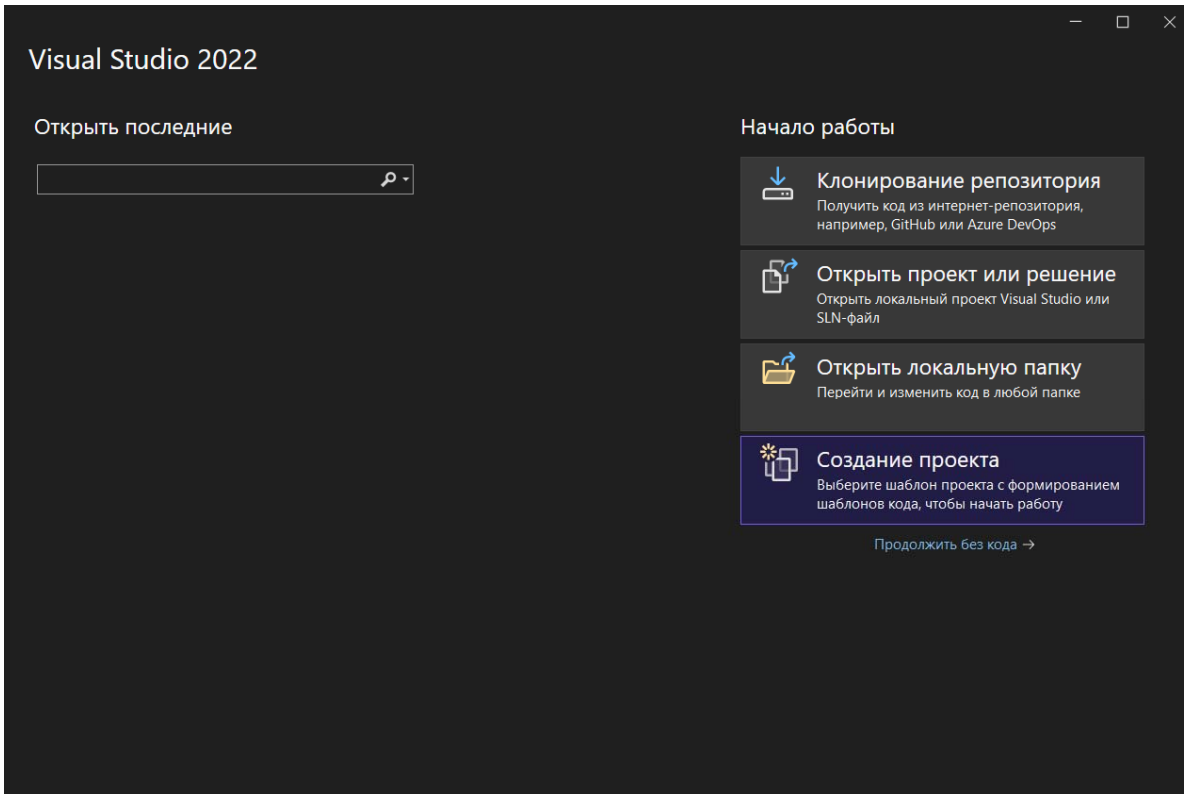


Рисунок 2.3 – Меню создание проекта

Следующем шагом в качестве типа создаваемого проекта указываем “Приложение Windows Forms (Майкрософт)”, как показано на рисунке 2.4.

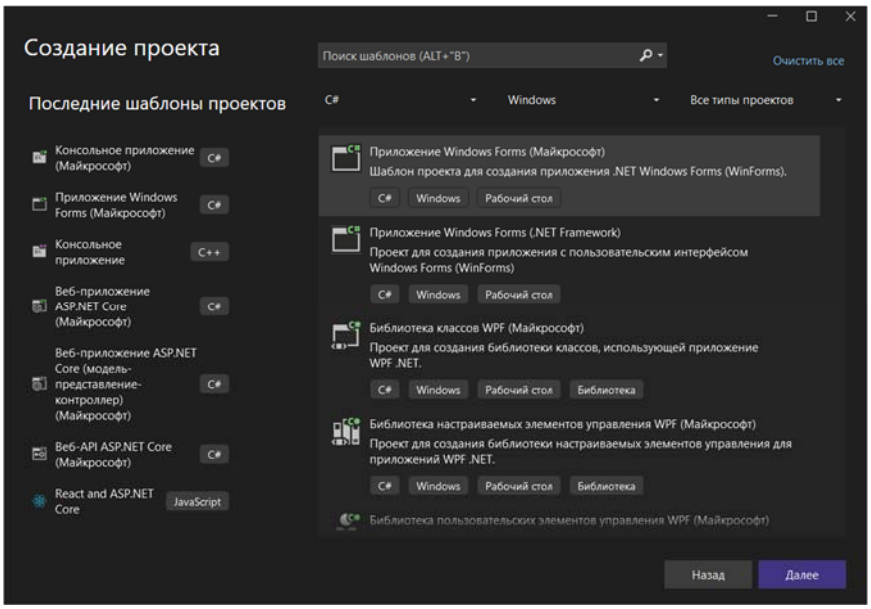


Рисунок 2.4 – Меню создание проекта

Далее указываем имя проекта и каталог, где будет располагаться проект, как показано на рисунке 2.5.

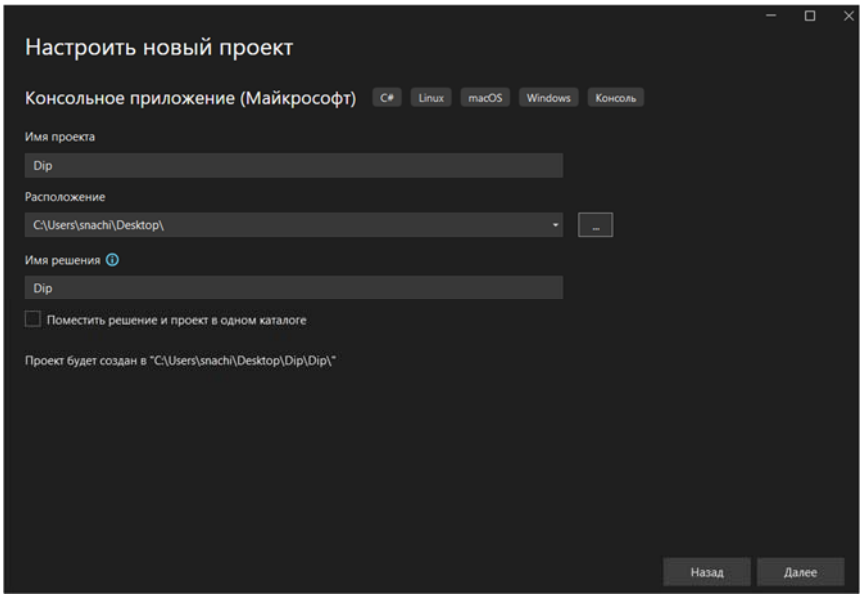


Рисунок 2.5 – Меню создание проекта

После этих действий будет собранно шаблонное приложение с использованием Windows Forms, как показано на рисунке 2.6. В нем мы можем приступать к разработке нашего приложения симулятора работы PIC микроконтроллера.

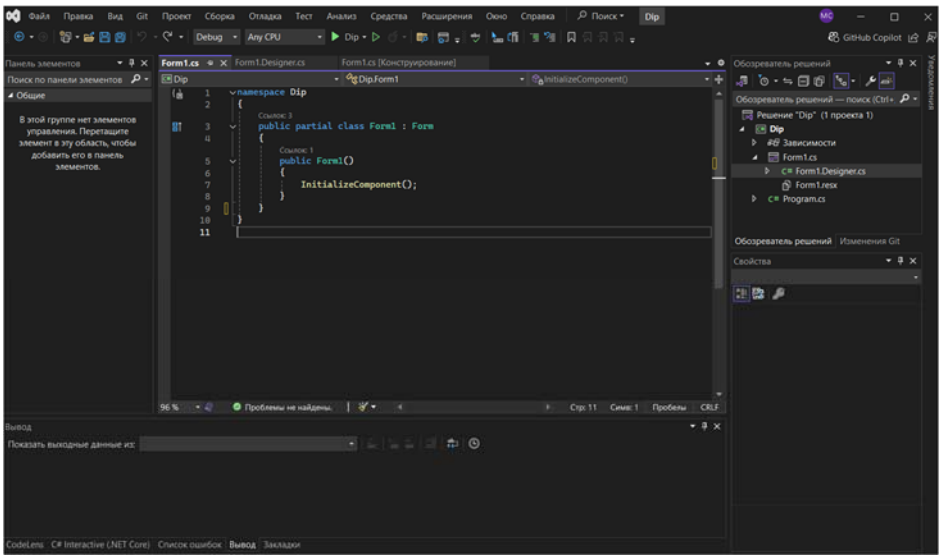


Рисунок 2.6 – Окно редактора программы

2.2 Описание структуры программы

Диаграмма классов, представленная на рисунке 2.7 помогает лучше понять необходимый к реализации функционал приложения. Ведь перед началом разработки программы необходимо определиться с используемыми в ней классами.

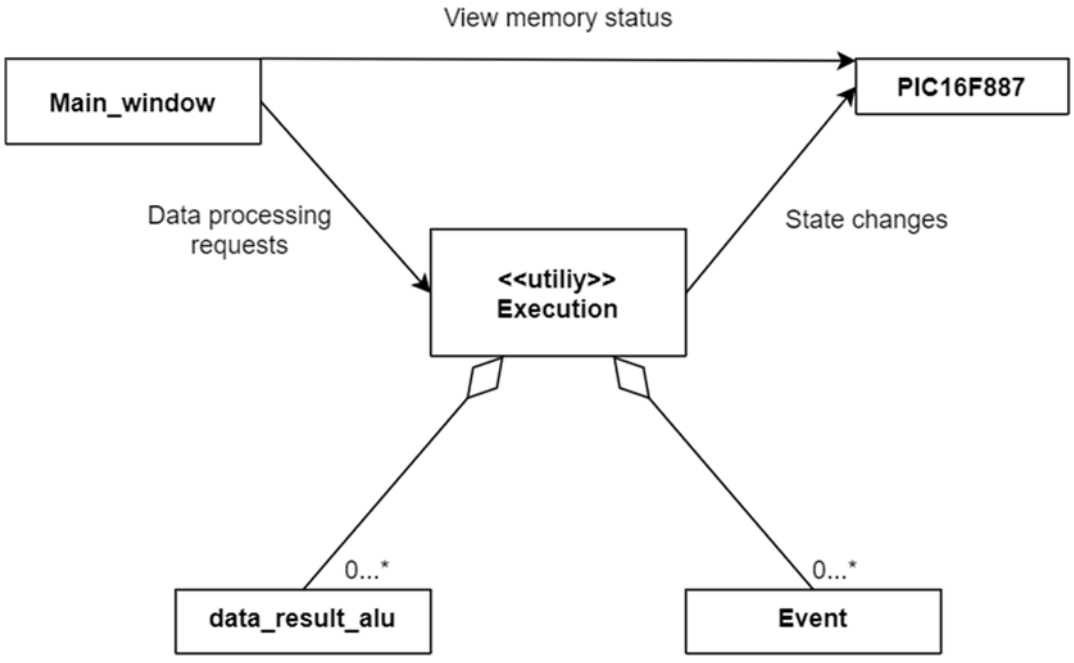


Рисунок 2.7 –UML диаграмма классов

Данное приложение будет разделено на 3 основных класса, в первом классе с названием Main_window будут определены основные методы работы программы, а также объявлены соответствующие формы, с которыми будет взаимодействовать пользователь, диаграмма класса Main_window представлена на рисунке 2.8.

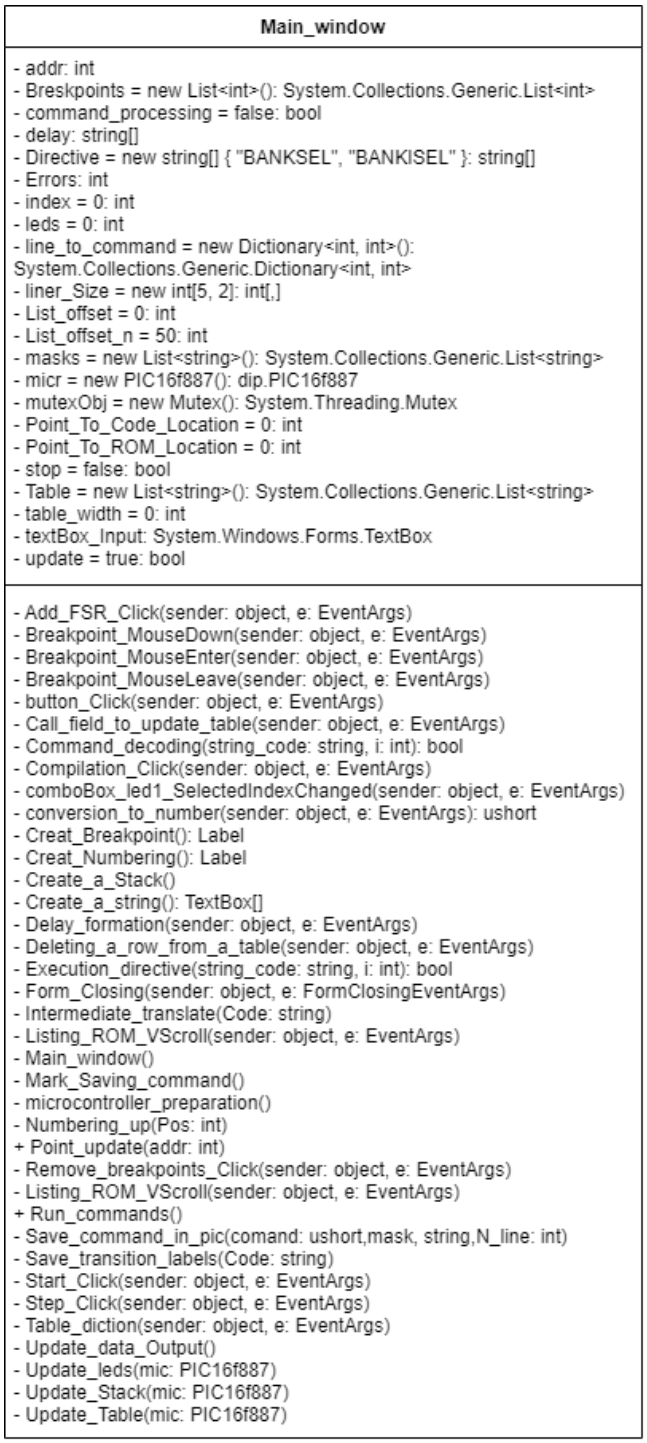


Рисунок 2.8 –UML диаграмма класса Main_window

Следующий класс под названием Execution будет представлен в виде методов, отвечающих за исполнение программы. Здесь все методы направлены на программное изменение состояния микроконтроллера. Также в этом классе используются два других небольших класса для реализации различных функций, диаграмма класса Execution представлена на рисунке 2.9.

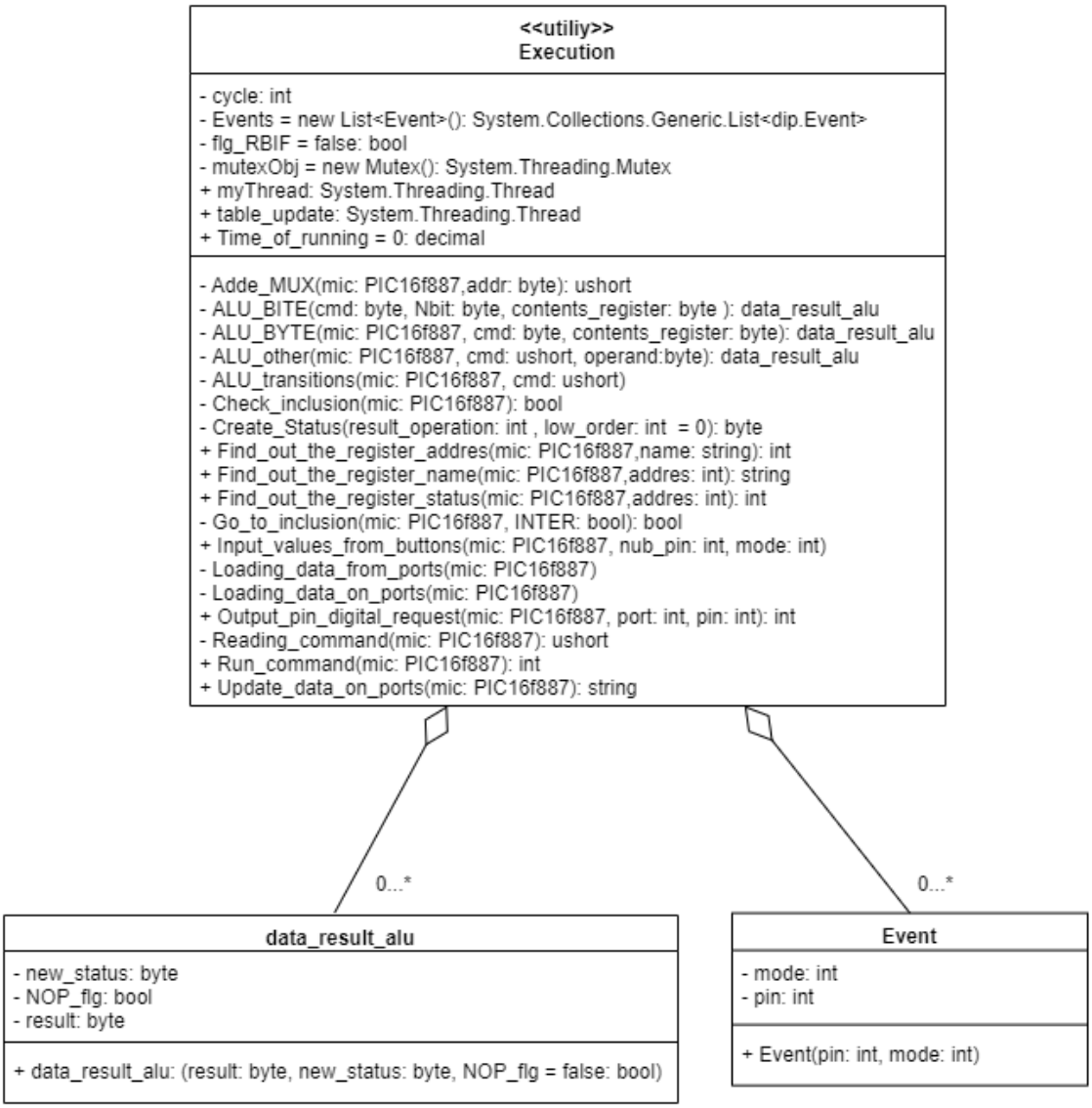


Рисунок 2.9 –UML диаграмма класса Execution

Последний класс с именем PIC16f887 имитирует поведение памяти PIC микроконтроллера. А также выполняет различные низшие функции, например работа таймеров и счетчиков. Диаграмма класса PIC16F887 представлена на рисунке 2.10.

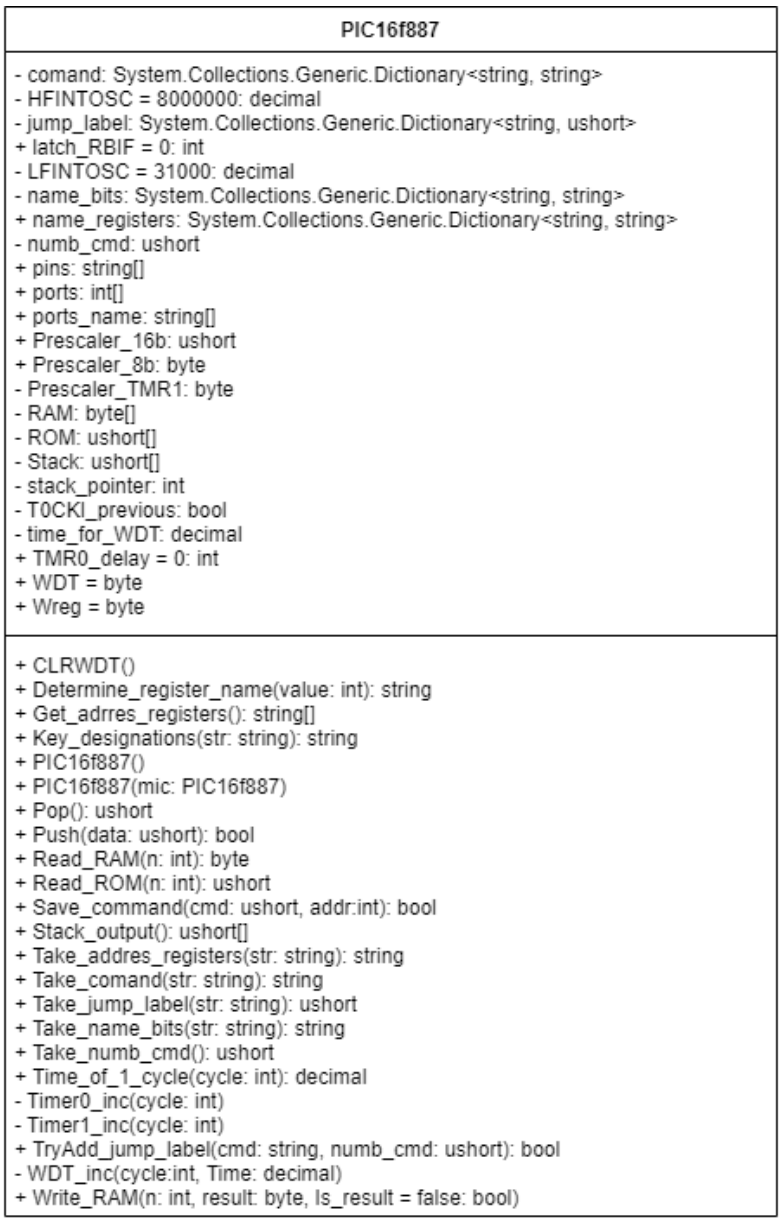


Рисунок 2.10 –UML диаграмма класса PIC16f887

2.3 Разработка программы

При разработке приложения небольшие вопросы были решены с помощью сайта metanit.com [6] и форума ci-sharp.ru [7].

2.3.1 Разработка компилятора

Первым этапом разработки была выбрана разработка интерфейса компилятора, чтобы на основании составленного машинного кода была возможность продолжить разработку. Для начала нам необходимо выбрать все

нужные для работы программы формы из списка, представленного программой Windows Forms. Для реализации компилятора нам нужны будут следующие компоненты:

– 2 формы RichTextBox для набора программы пользователем и для консоли вывода;

– форма Button для запуска компиляции;

– форма Panel для отображения нумерации строк;

– форма Label в количестве 64 штуки для динамического отображения номеров строк.

Для каждого компонента выставляем необходимые параметры и назначаем методы на необходимые события.

При добавлении новой строки в поле ввода программы сразу добавляется ее номер, если в программе появляется необходимость прокрутки, то формы Label, находящиеся в форме Panel, прокручиваются вместе с основной программой. Если номер строки выйдет за поле прорисовки на экране на определенное значение, то он меняет свою позицию и свой номер таким образом, чтобы у пользователя складывалось впечатление бесконечной прокрутки. Динамичная нумерация строк не требует большого количества процессорного времени и большого количества памяти на свою прорисовку.

При нажатии на кнопку “Compilation” у нас запускается метод Compilation_Click(), оторый блокирует возможность редактирования поля для ввода программы. Далее происходит подготовка необходимых поле для дальнейшей работы с помощью метода microcontroller_preparation(), и уже после этого начинается обработка программы. Весь документ проверяется построчно, вызывается метод Save_transition_labels() в котором происходит на поиск одиноких слов, не являющимися директивами или командами, после чего они сохраняются в коллекцию переходов в программе, сопоставляя название метки с ее адресом. Если будет найдено повторение одной и той же метки, то буде пользователю выведена ошибка. Далее в методе Intermediate_translate().

программа повторно проходит по каждой строке и проверяет на наличие команд и директив. При нахождении команды вызывается метод Command_decoding() на место которых надо вставить определенную команду, проверяется название команды, и в соответствии с названием определяется ее набор параметров через метод conversion_to_number(), параметры могут быть представлены в виде специальных слов, а именно названия регистра или названия бита в регистре, в виде двоичной записи, десятичной записи, восьмеричной записи, шестнадцатеричной записи и в виде адреса метки. При нехватке параметров команды или невозможности их распознать пользователю будет выведено соответствующие сообщение об ошибки. При нахождении имени директивы происходит ее выполнение в методе Execution_directive(). Также каждая с помощью метода Save_command_in_pic() команда сразу конвертируется в машинный код и сохраняется соответствие ее порядкового номера и строки в написанной программе. Эта информация пригодиться для отладки. При отсутствии ошибок отображается и подсвечивается машинный код в соответствующем поле, которое отображает состояние Program Memory памяти.

2.3.2 Разработка отладчика

Следующим этапом стала разработка методов, необходимых для запуска расшифровки и выполнения команд, а также обновления выводимых параметров состояния. Выбираем из списка форм компоненты из списка:

– форма RichTextBox для отображения состояния Program Memory;

– 3 формы Button для различных задач;

– 3 форма Panel для отображения границ отображения элементов;

– 2 формы Label для отображения отметок следующей выполняемой команды;

– 64 формы Label для отображения точек остоновки;

– форма Label для отображения текущего времени выполнения программы;

– форма TextBox для ввода задержки;

– 2 формы GroupBox для отображения состояний RAM и стека.

Внутри формы GroupBox отвечающей за отображения состояния RAM также располагаются следующие компоненты:

– форма ComboBox выбора добавляемого регистра;

– форма Button добавления выбранного регистра;

– 5 форм Label для отображения названий заголовков;

– форма FlowLayoutPanel для отображения параметров регистров.

В форме GroupBox отвечающей за отображения состояния стека будут использованы следующие компоненты:

– 2 формы Label для отображения названий заголовков;

– форма FlowLayoutPanel для отображения уровней стека.

После успешной компиляции программы пользователь может начать ее отладку, для начала он может выбрать необходимые ему регистры с помощью таблицы просмотра состояний. В таблице отображаются адрес в шестнадцатеричной системе счисления, имя регистра, а также его значение в шестнадцатеричном, двоичном и десятичном представлениях.

Пользователь может кликнув двойным щелчком на ячейку в столбцах адреса или названия регистра под соответствующими заголовками, это вызовет метод Call_field_to_update_table() который создаст новую форму TextBox на том же месте и направит на нее фокус. При покидании созданной формы происходит вызов метода Table_diction(), который осуществляет поиск соответствующего регистра или области в памяти микроконтроллера. При успешном нахождении в таблицу будет добавлена новая строка, а созданная форма TextBox будет уничтожена.

Также есть возможность добавить строку в таблицу через выбор именного регистра или области в памяти микроконтроллера с помощью формы ComboBox. При нажатии на кнопку “Add_FSR” вызывается метод Add_FSR_Click(), который так же добавляет строку с выбранным реестром в таблицу.

При выборе любой ячейки в таблице можно нажать клавишу Delete на клавиатуре и тогда будет вызван метод Deleting_a_row_from_a_table(), с помощью которого вся строка будет убрана из таблице.

В окне приложения располагается окно с отображением стека микроконтроллера, он имеет всего 8 уровней и в этом окне только отображается значения уровней стека.

Обновление таблиц состояний RAM и стека происходит автоматически с использованием другого потока, отвечающего за обновление данных о микроконтроллере, так что основному потоку это не помешает.

Для обработки команд построчно пользователь может использовать кнопку “Step_Click”, при нажатии на которую поочередно запускаются методы Run_command() и Point_update(). Первый метод выполняет обработку следующей команды, второй метод обновляет позицию указателей на текущую исполняемую команду.

Для обработки команд циклически пользователю необходимо нажать на кнопку “Start”, и тогда блокируется часть возможностей до завершения цикла пользователя во избежание возможных ошибок. Затем запускается поток, который выполняет примерно те же действия что и кнопка “Step_Click”, но уже учитывая критические секции и выполняя команды с определенной задержкой, указанной в текстовом поле вверху почти по центру экрана. Данный цикл будет остановлен либо при достижении любой точки остановки, либо при нажатии на кнопку “Stop”.

Точки остановки пользователь сможет расставить напротив любой строки кода, так же они подсвечиваются при наведении на них мышкой для более простого взаимодействия. Если на точку остановки нажать повторно, она будет удалена. Данные функции реализуются с помощью методов Breakpoint_MouseDown(), Breakpoint_MouseEnter() и Breakpoint_MouseLeave(). При желании все точки остановки можно убрать с помощью кнопки “Remove bp”, которая вызывает соответствующий метод Remove_breakpoints_Click().

2.3.3 Разработка периферии

Данный модуль будет доступен пользователю также при успешной компиляции, однако работу периферии можно узнать только после исполнения микроконтроллером команды. На текущий момент для реализации компонентов периферии нам понадобятся следующие формы:

– две формы GroupBox для отображения соответствующих элементов приферии;

– 5 форм ComboBox выбора контактов микросхемы;

– 4 формы Button для подачи цифровых сигналов;

– 8 форм PictureBox для имитации светодиодов.

В GroupBox, отвечающем за отображение светодиодов с помощью элемента ComboBox можно выбрать любую из представленных линеек контактов микросхемы для подключения к ним светодиодов. Обновление отображаемой информации на светодиодах происходит автоматически в отдельном потоке, отвечающем за обновление данных о микроконтроллере через метод Update_leds().

В другом GroupBox можно подключить кнопки на любой контакт микросхемы и при нажатии на кнопку вызывается метод button_Click(), который создает событие цифрового ввода на контакт через метод Input_values_from_buttons(), данное событие будет обработано перед выполнением следующей команды.

2.3.4 Разработка обработчика команд

В данном пункте будет рассказано ту часть программы, которая отвечает за исполнение команд микроконтроллером. Этот этап является сердцем приложения симулятора работы PIC микроконтроллера. Основным методом отвечающим за данную часть программы является Run_command(). В нем первым делом проверяются все события ввод, произошедшие на контактах, и проверяется вводимая информация, так же выставляются флаги произошедших прерываний. Далее идет проверка, в спящем ли состоянии находится микроконтроллера, если

да, то проверяются события вывода из спящего режима и на этом обработка команды заканчивается, иначе идет дальнейшая обработка команды.

В начале нам необходимо считать команду, на которую указывает СК, после чего мы должны его увеличить. Затем происходит выборка команды, команда исполняется либо в отдельно части программы, либо в общем блоке команд.

Если команда принадлежит к байтовым операциям, то в начале вызывается метод Adde_MUX() определяющий ячейку в памяти, данный метод учитывает как прямую адресацию так и косвенную. Затем прочитывается ячейка в памяти по указанному адресу и будет вызван метод ALU_BYTE() который совершает байтовую операцию. После данных действий происходит обновление данных регистров микроконтроллера. Походим образом работает обработка битовых операций, за исключением того, что здесь используется метод ALU_BITE() и другой формат записи данных в микроконтроллер. Команды относящиеся к приходам исполняются в методе ALU_transitions(), здесь СК изменяется на необходимое значение, а также при необходимости используется стек для запоминания адреса возврата. Команды, в который задействованы константы выполняются в методе ALU_other(), после результат будет записан только в регистр Wreg.

После обработки команды производиться увеличения внутренних таймеров и расчет времени работы микроконтроллера, относительно выбранной частоты работы микропроцессора. Так же при переполнении таймеров будет поднят бит соответствующего прерывания.

Далее происходит проверка всех прерываний с помощью метода Check_inclusion(), которые возникли после всех предыдущих действий. И если биты разрешения прерывания так же установлены, то происходит обработка возникшего прерывания в методе Go_to_inclusion().

Завершающим действием происходит обновление выводимых данных на контакты микроконтроллера, в вызванном методе Loading_data_on_ports().

2.3.5 Разработка класса микроконтроллера

В данном классе был сымитировано поведение памяти микроконтроллера в соответствии с документацией [5]. Так же данный класс имеет конструкторы, которые имитируют работу при первом запуске микроконтроллера и при его сбросе, а также различные методы чтения и записи в память. Здесь находятся списки используемых обозначений такие, как названия регистров и битов и их адреса в памяти.

2.4 Описание доступных функций

В приложении разделено на несколько частей, с которыми пользователь может взаимодействовать, первым шагом ему будет необходимо написать программу, которая будет конвертирована в машинный код. С помощью данного функционала обучающийся может набирать код программы, редактировать его, проверять его на наличие ошибок при нажатии на кнопку компиляции, просматривать допущенные ошибки, а также расставлять комментарии после спецсимвола, итогом работы данного интерфейса будет составленный машинный код микроконтроллера.

Вторым шагом для пользователя будет взаимодействие с обработчиком команд с возможностями покомандной, и циклической обработки. В данном интерфейсе обучающийся сможет расставлять точки остановки, по достижении которых циклическая обработка команд будет прекращаться. Так же он сможет регулировать скорость выполнения команд под удобный для него уровень восприятия. Следующая команда, которая должна быть выполнена, помечается специальным указателем и в написанном, и в машинном кодах, что поможет сопоставить машинный код с программным, так же в машинном кода биты отвечающие за разные функции окрашены в соответствующий цвет. Пользователь так же сможет просматривать всю внутреннюю память контроллера для лучшего понимания происходящих в нем операций.

Одновременно с обработчиком команд пользователю предоставляется возможность подавать различные сигналы в микроконтроллер через его периферию, а также просматривать сигналы, приходящие с него.

2.1.1. Описание работы компилятора

При первом открытии пользователю будет представлено следующие окно, как показано на рисунке 1, в котором пользователь сможет набрать, редактировать, настраивать свою программу как показано на рисунках 2.11 и 2.12.

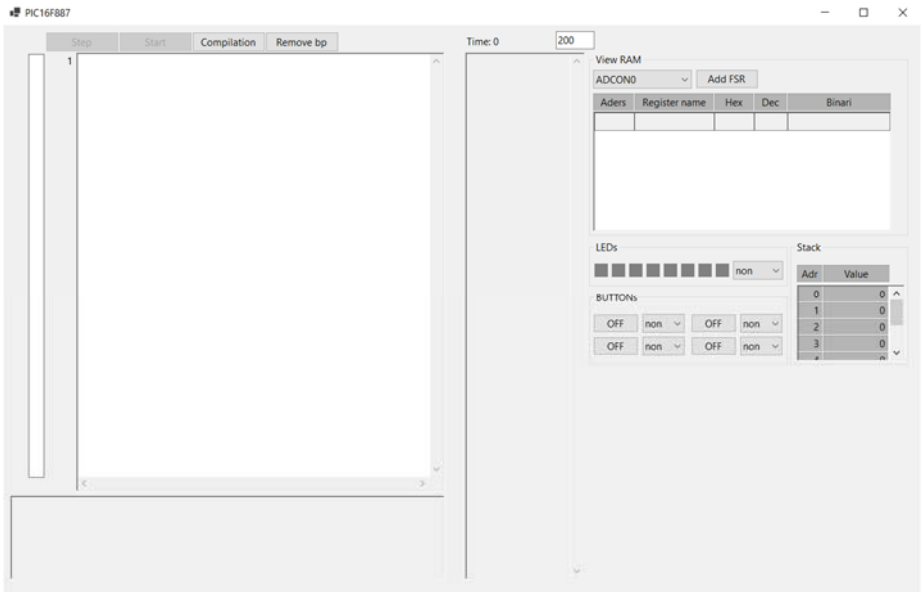


Рисунок 2.11 – Программа при первом открытии

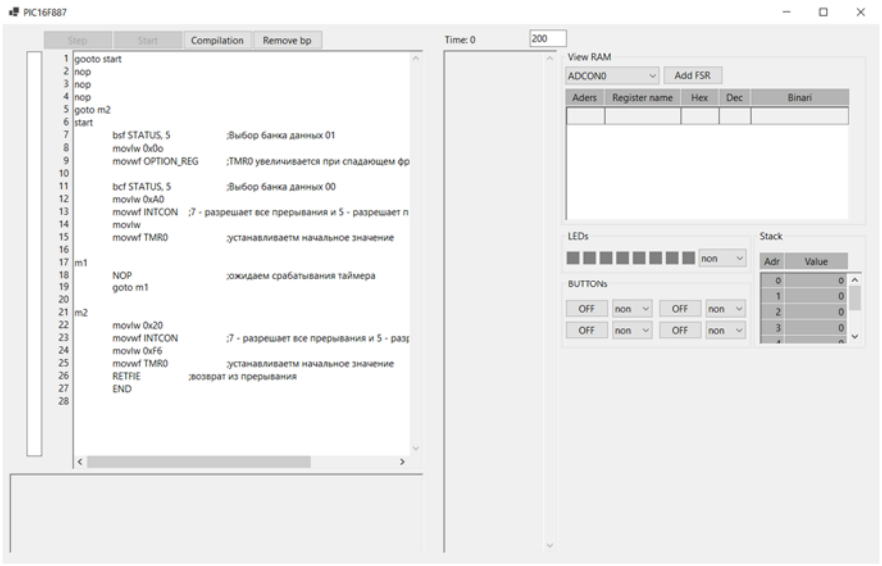


Рисунок 2.12 – Окно редактора программы

В любой момент написания программы пользователь сможет проверить ее на ошибки нажав на кнопку “Compilation”. При нахождении ошибок программа будет выводить номер строки, в которой она допущена и краткое описание данной ошибки как показано на рисунках 2.13 и 2.14. Это так же необходимо для компиляции программы, в случаи отсутствии ошибок.

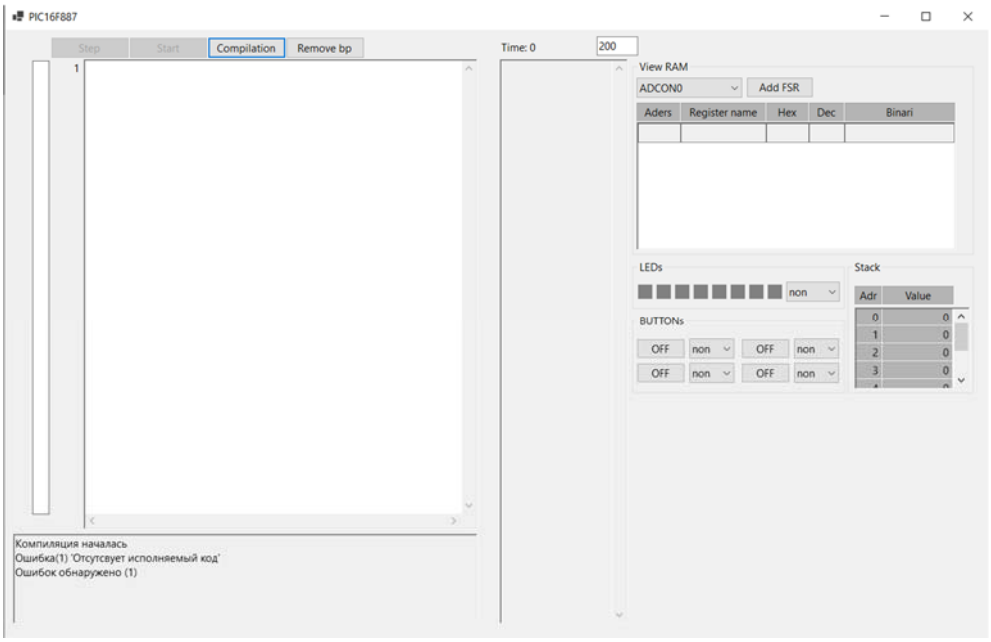


Рисунок 2.13 – Окно редактора программы

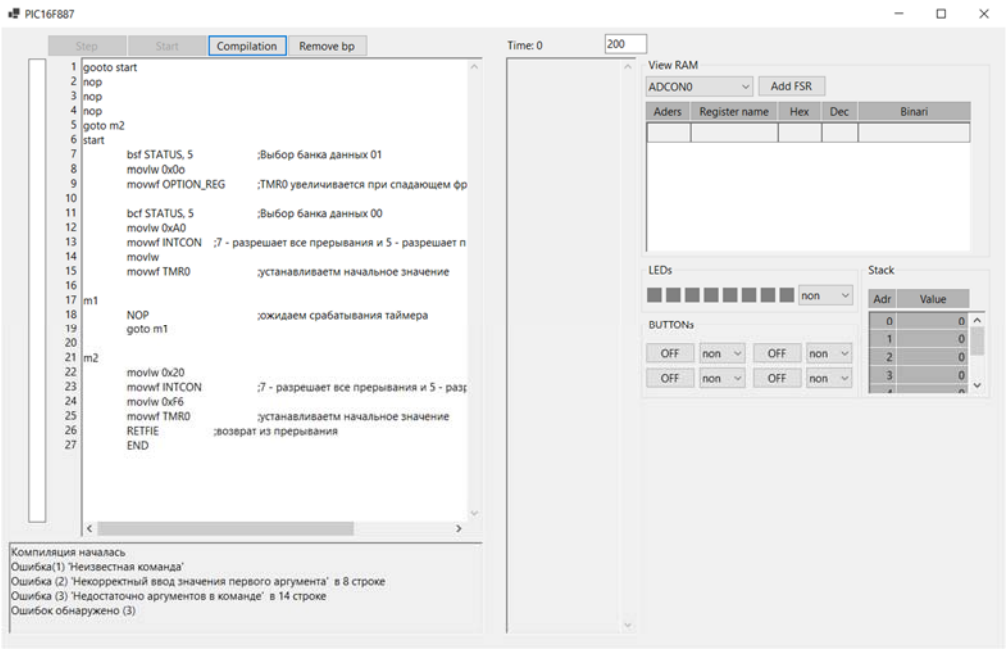


Рисунок 2.14 – Окно редактора программы

Таким образом пользователю необходимо исправлять свою программу до тех пор, пока не будут устранены все ошибки. После чего приложение сможет составить машинный код, который будет загружен в память симулированного PIC микроконтроллера. Так же на данном этапе появляются указатели на следующую выполняемую программу как около написанного кода, так и около машинного, становятся доступными кнопки “STEP” и “Start”, а кнопка “ Compilation” заменяется на “Stop debug” как показано на рисунке 2.15.

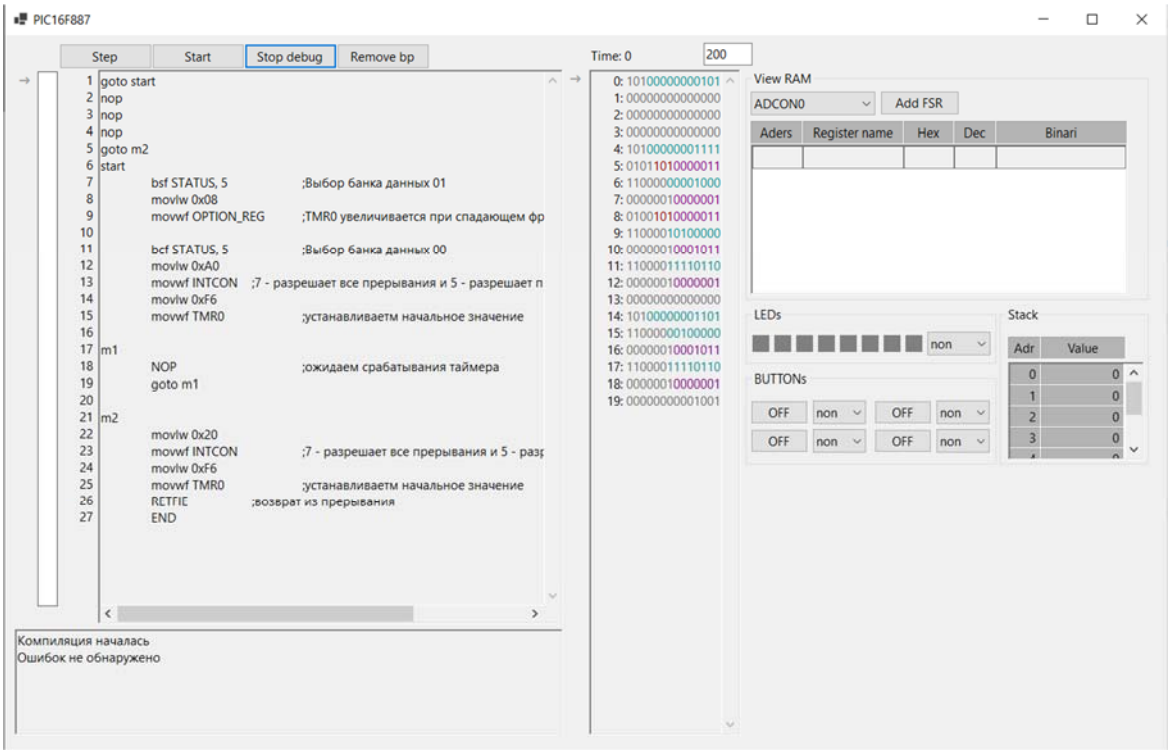


Рисунок 2.15 – Окно редактора программы

На данном моменте взаимодействие с компилятором оканчивается, и пользователь может перейти к следующему этапу изучения работы PIC микроконтроллера.

2.1.2 Описание работы отладчика

При успешной компиляции программы пользователь сможет настроить для себя просмотр необходимых для него регистров, это можно сделать любы из представленных способов способами. Пользователь может вписать название необходимого регистра в поле “Register name”, как это показано на рисунках 2.16

и 2.17 или вписать адрес необходимого регистра в поле “Adres”, как это показано на рисунке 2.18, также в данном поле ввода есть список из доступных регистров, пользователь может выбрать любой регистр имеющий свое имя из списка и добавить его по нажатию на кнопку “Add FSR” как показано на рисунках 2.19 и 2.20.

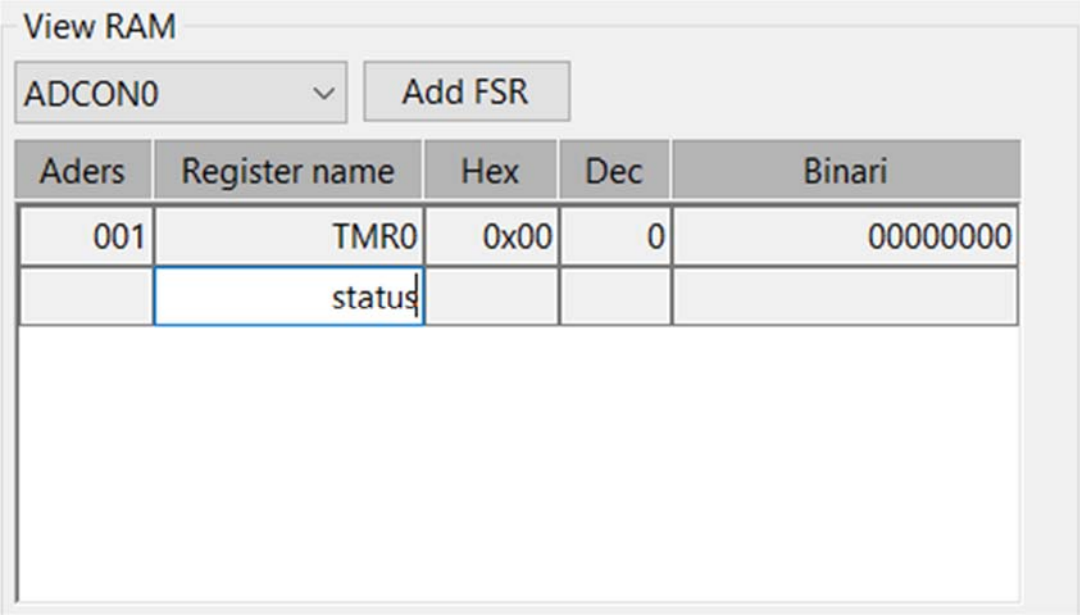


Рисунок 2.16 – Ввод имени регистра в поле “Register name”

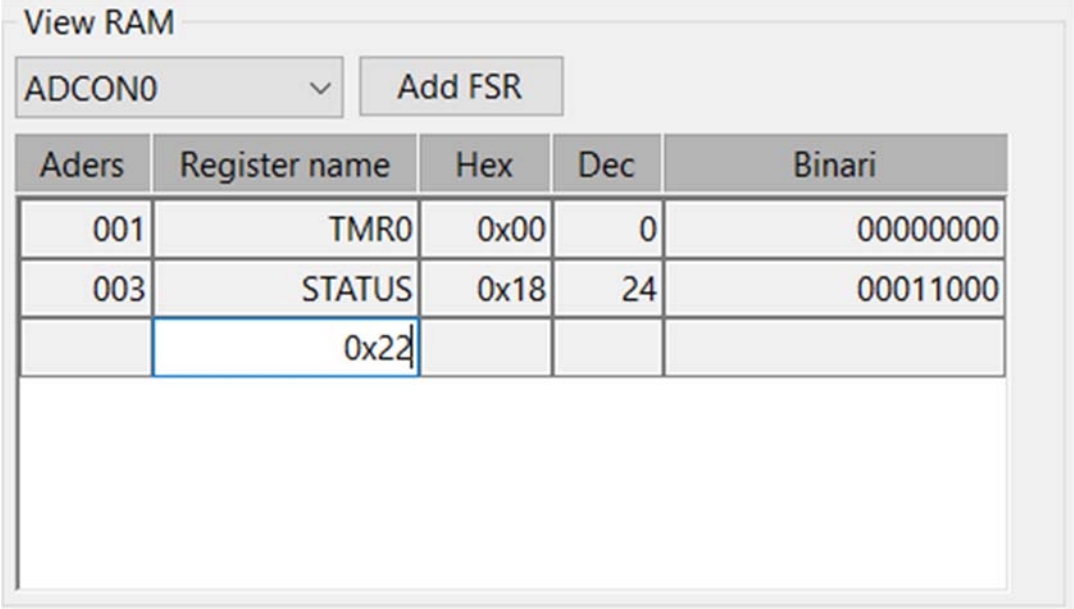


Рисунок 2.17 – Ввод имени регистра в поле “Register name”

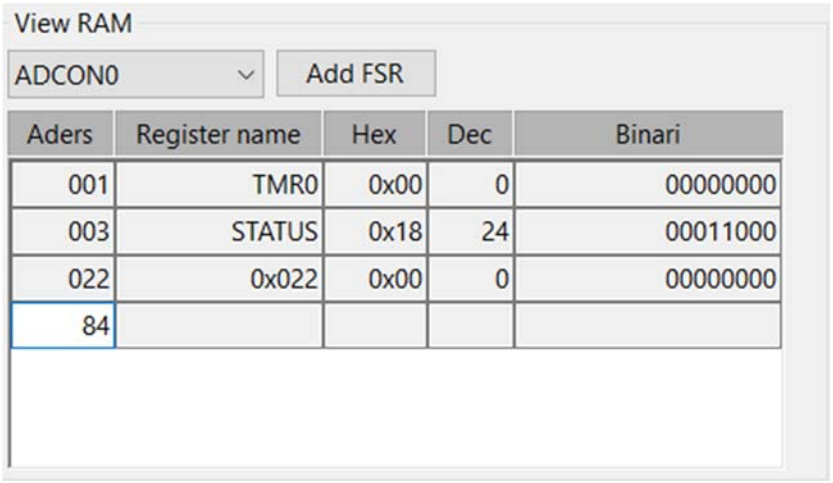


Рисунок 2.18 – Ввод адреса регистра в поле “Adres”

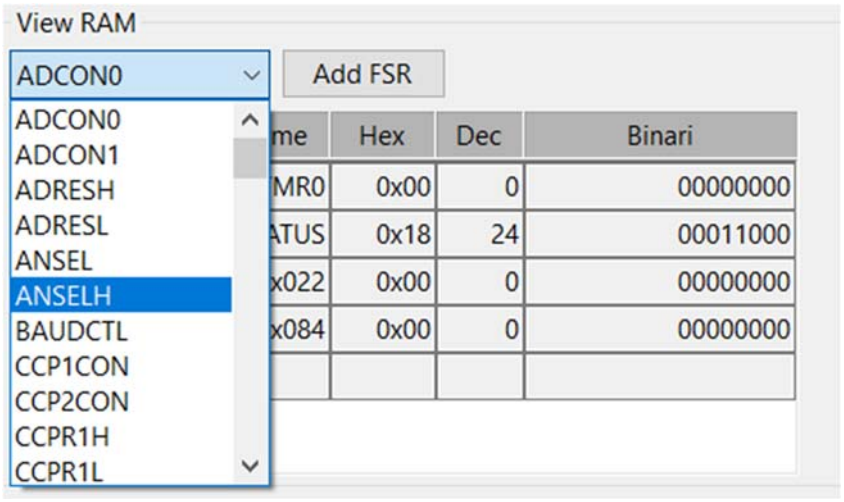


Рисунок 2.19 – Добавление регистра через список

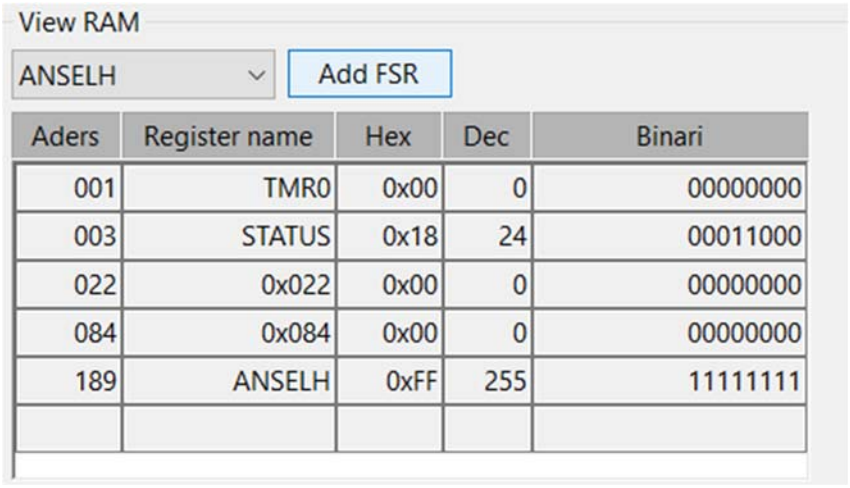


Рисунок 2.20 – Добавление регистра через список

Если же необходимость наблюдения за некоторыми регистрами пропадет, то пользователь может удалить его, выделив необходимую строку, путем одиночного нажатия на любое поле в нем левой кнопки мыши и затем нажав на клавишу “Delete” на клавиатуре, как показано на рисунке 2.21.

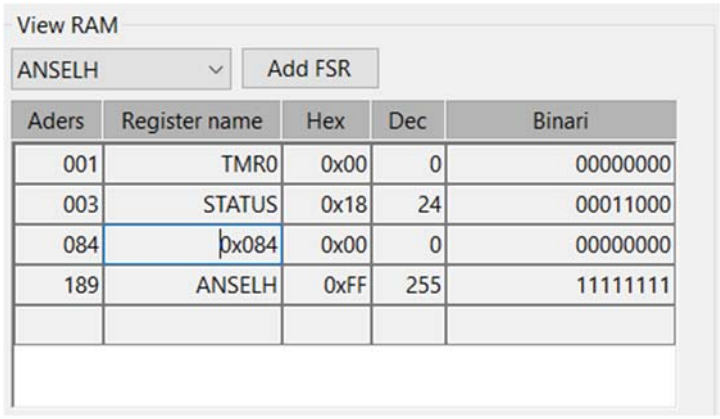


Рисунок 2.21 – Удаление регистра из списка

Хоть в реальном PIC микроконтроллере стек отделен от основной памяти пространства и отсутствуют какие-либо способы прямой записи и чтения в него, в арсенале у пользователя для обучения его содержимое все же будет представлено, как показано на рисунке 2.22.

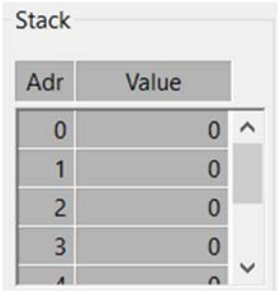


Рисунок 2.22 – Просмотр стека

Для того чтобы обработать следующую на очереди команду пользователь может нажать на кнопку “Step”, после чего все данные, на которые влияет команда будут обновлены. Указатель на ткущую команду изменит свое положение, время работы контроллера увеличиться, затрагиваемые регистры изменят свое значение, как показано на рисунке 2.23.

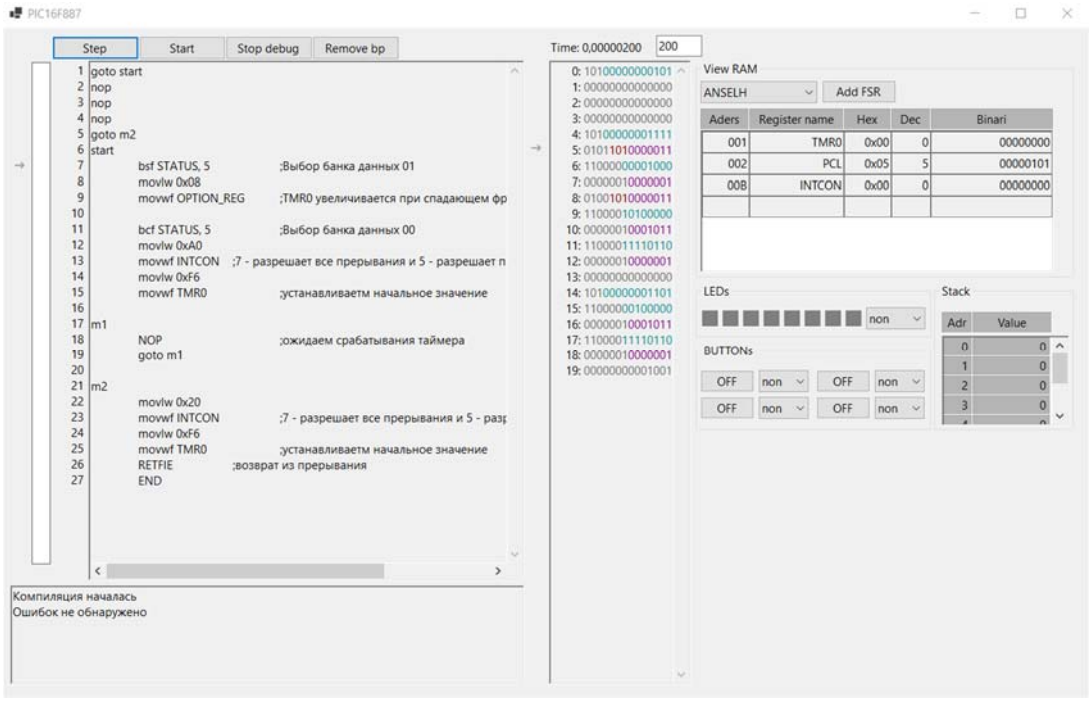


Рисунок 2.23 – Последствия нажатия на кнопку “Step”

Иным способом пользователь может нажать на кнопку “Start” и тогда команды будут обрабатываться циклически с выбранной задержкой, как показано на рисунке 2.24. Изначально значение задержки равно 200мс.

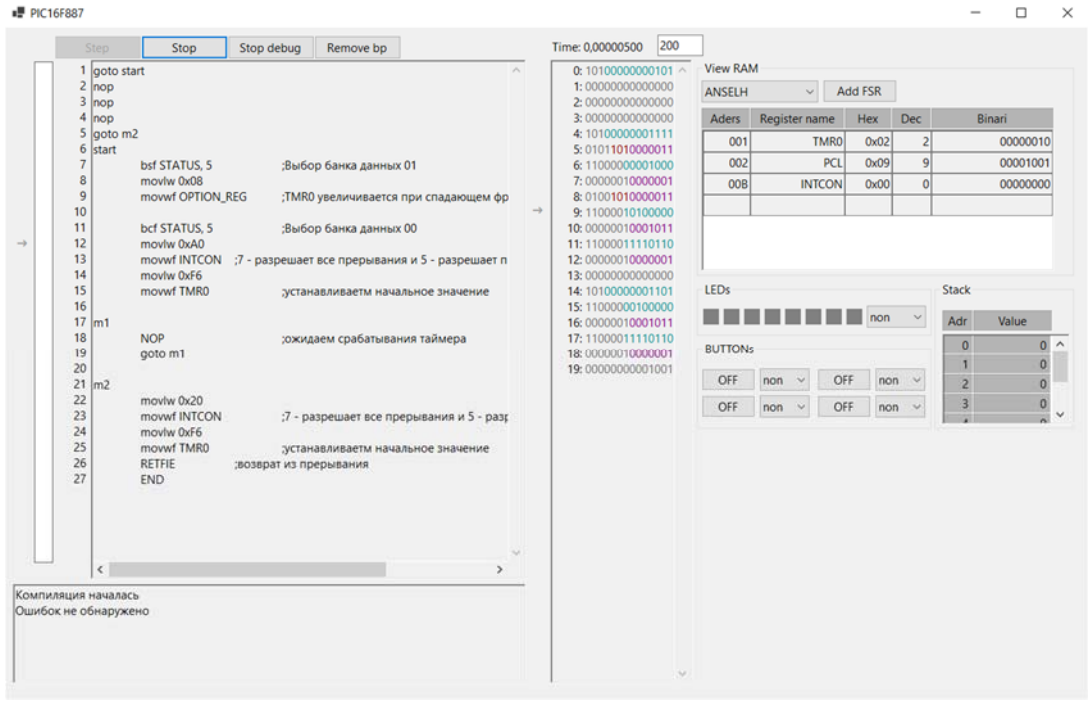


Рисунок 2.24 – Последствия нажатия на кнопку “Step”

При необходимости задержку можно изменить подходящего значения, как показано на рисунке 2.25.

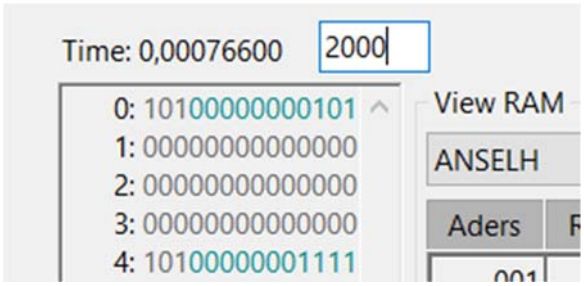


Рисунок 2.25 – Изменение задержки

При циклической обработке программы пользователь может использовать точки остановки, дойдя до любой из который программа остановиться, как показано на рисунке 2.26.

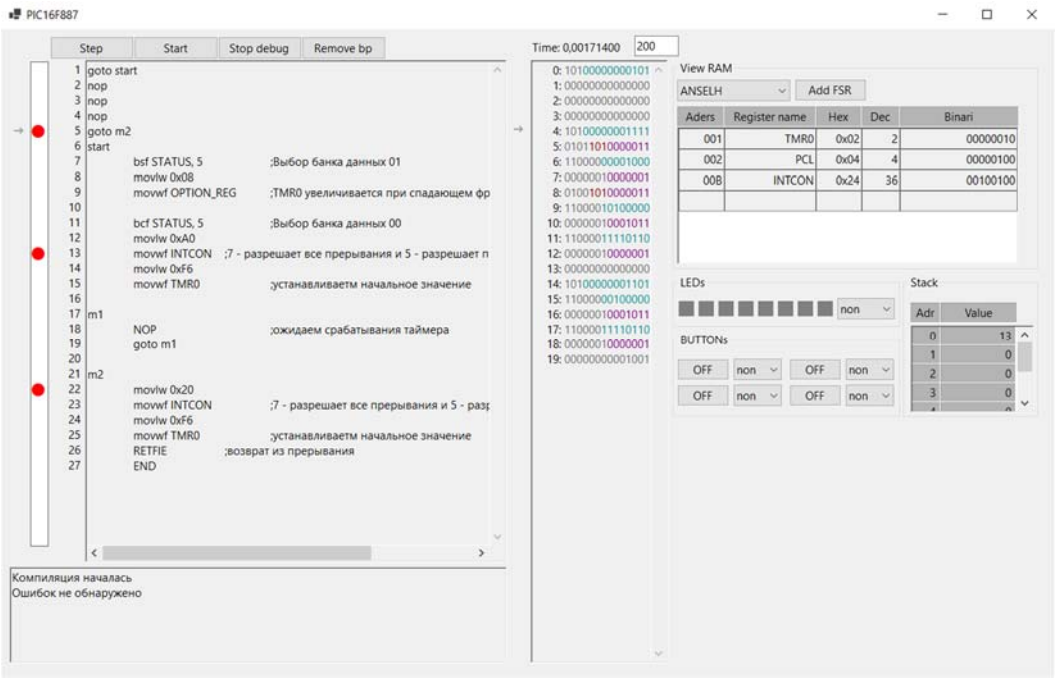


Рисунок 2.26 – Точки остановки

Если пользователю больше не нужны расставленные точки остановки, он может убрать из одним щелчком с помощью кнопки “Remove br”, как показано на рисунке 2.27.

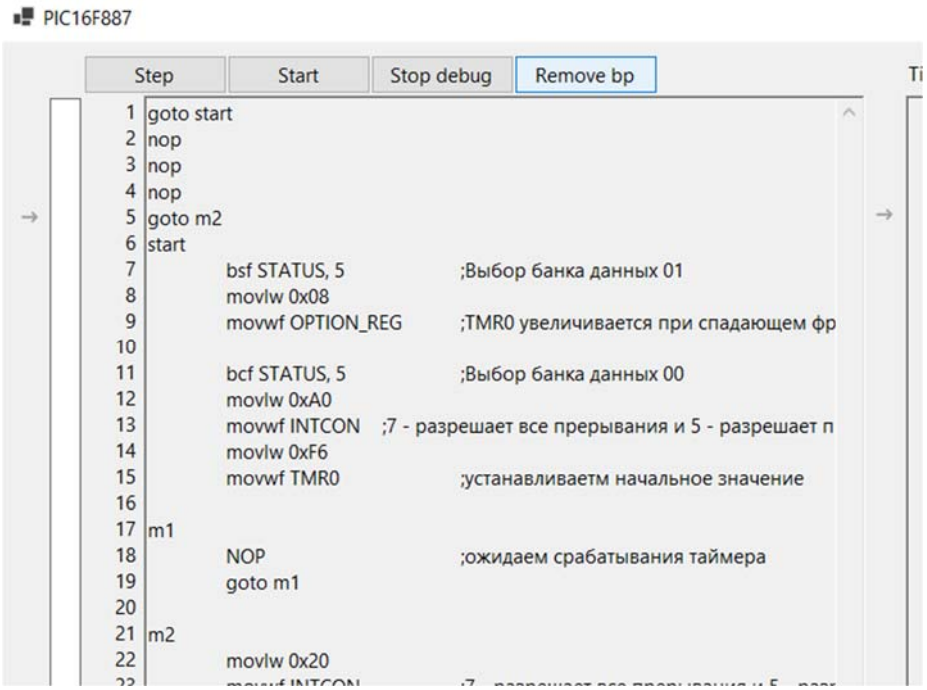


Рисунок 2.27 – Удаление точек остановки

2.1.3 Описание работы периферии

Одновременно с началом работы с отладчиком пользователь сможет взаимодействовать с интерфейсом периферии, на данный момент она представлена цифровым вводом и выводом в виде разных блоков.

Для вывода цифрового сигнала обучающийся может назначить любой из доступных портов, на подключение к линейке светодиодов, как показано на рисунке 2.28.

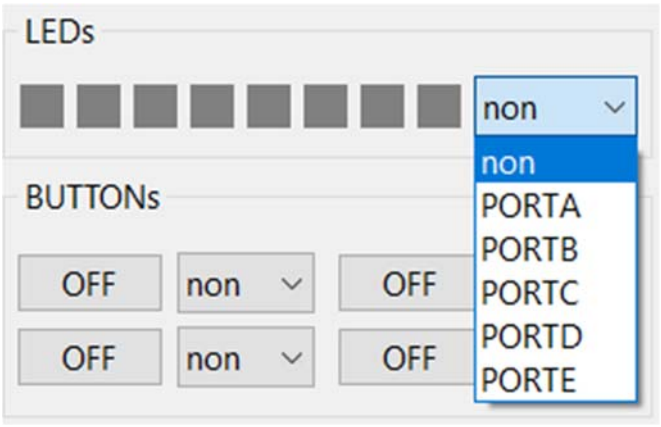


Рисунок 2.28 – Выбор порта цифрового вывода

Если соответствующие порты микроконтроллера не назначены на цифровой вывод, то цвет светодиодов будет таким же серым, как показано на рисунке 2.29.

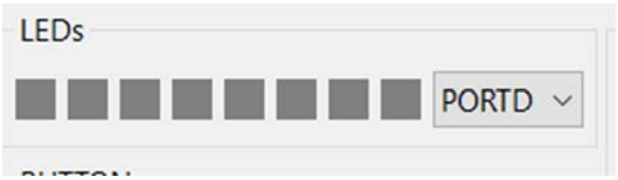


Рисунок 2.29 – Недоступность порта вывода

После внутренней настройки микроконтроллера через программу, полоска светодиодов начнет отображать выводимые данные, как показано на рисунке 2.30.

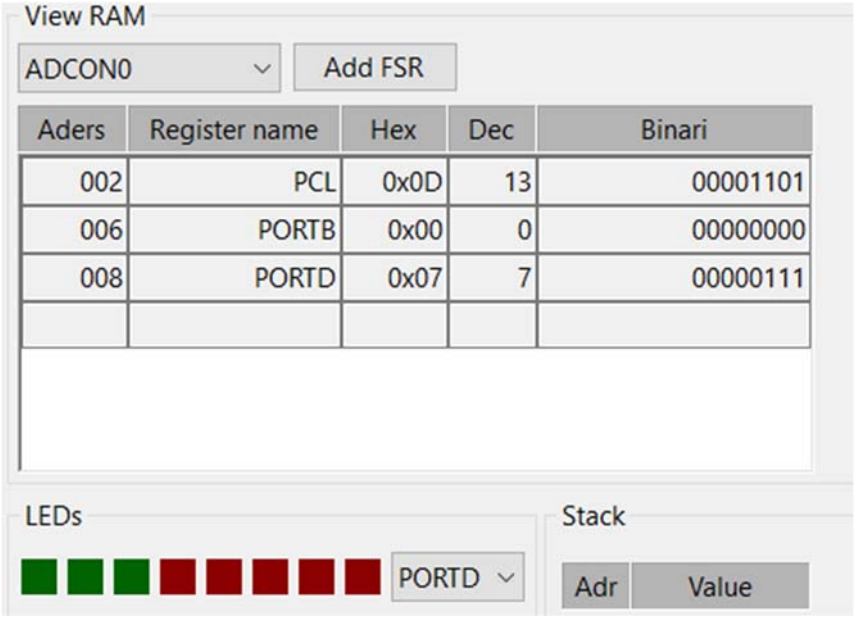


Рисунок 2.30 – Работа порта вывода

Если порт доступен лишь частично, то ненастроенная на вывод часть светодиодов будет окрашена в серый цвет, как показано на рисунке 2.31.

Рисунок 2.31 – Частичная работа порта вывода

Для подачи цифрового сигнала на порт микроконтроллера пользователь может выбрать любой порт для каждой кнопки, как показано на рисунке 2.32.

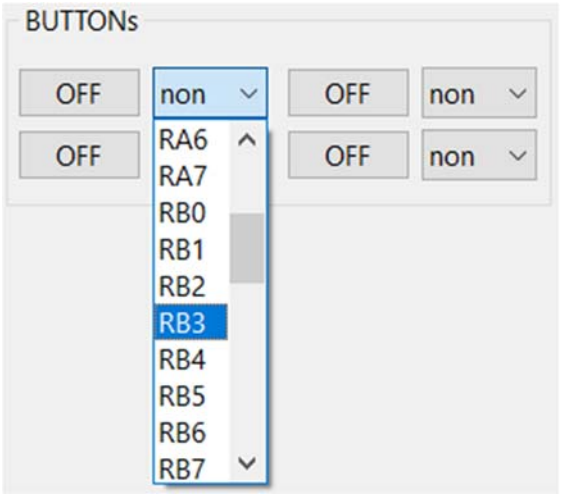


Рисунок 2.32 – Частичная работа порта вывода

После чего пользователь сможет отправить цифровой сигнал на соответствующий контакт на микроконтроллер, но входящее значение будет зарегистрированы ли в начале следующего исполнения команды, как показано на рисунке 2.33.

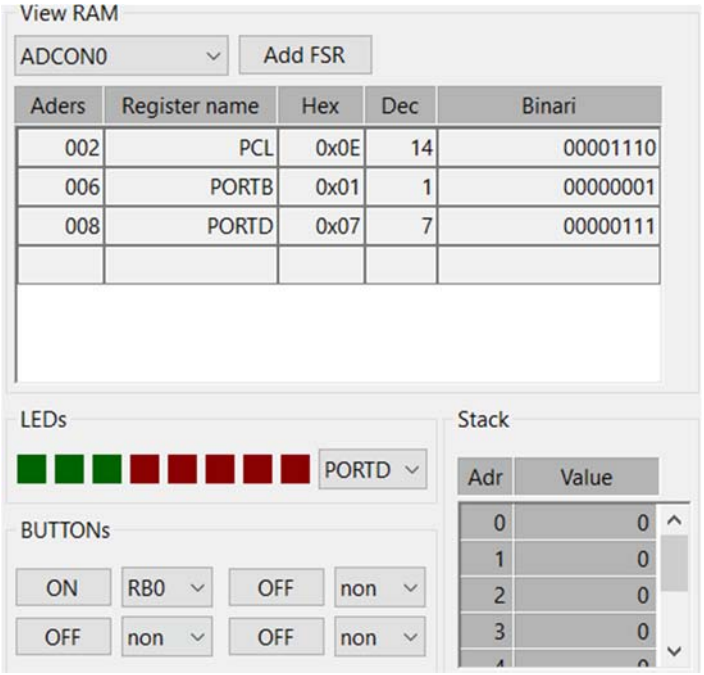


Рисунок 2.33 – Цифровой ввод

Однако если выбранный контакт пока еще не настроен на выход, то будет выведено сообщение об ошибке для устранения допущенной ошибки, как показано на рисунке 2.34.

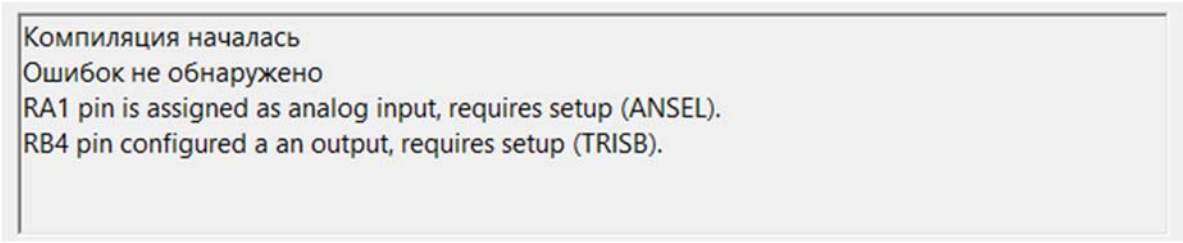


Рисунок 2.34 – Сообщение об ошибке

В итоге можно заключить, что на данный момент программа в достаточной степени предоставляет возможности для обучения начинающих специалистов, и в достаточной степени симулирует возможности микроконтроллера PIC16F887.

3 Анализ результатов

3.1 Результаты разработки приложения симулятора

В результате выполнения данного технического задания был реализован приложение симулятор работы микроконтроллера PIC16F887 с монолитной архитектурой. Приложение было написано на языке C# с использованием технологии Windows Forms. Написанное приложение работает по следующему алгоритму, отображенному на рисунке 3.1.

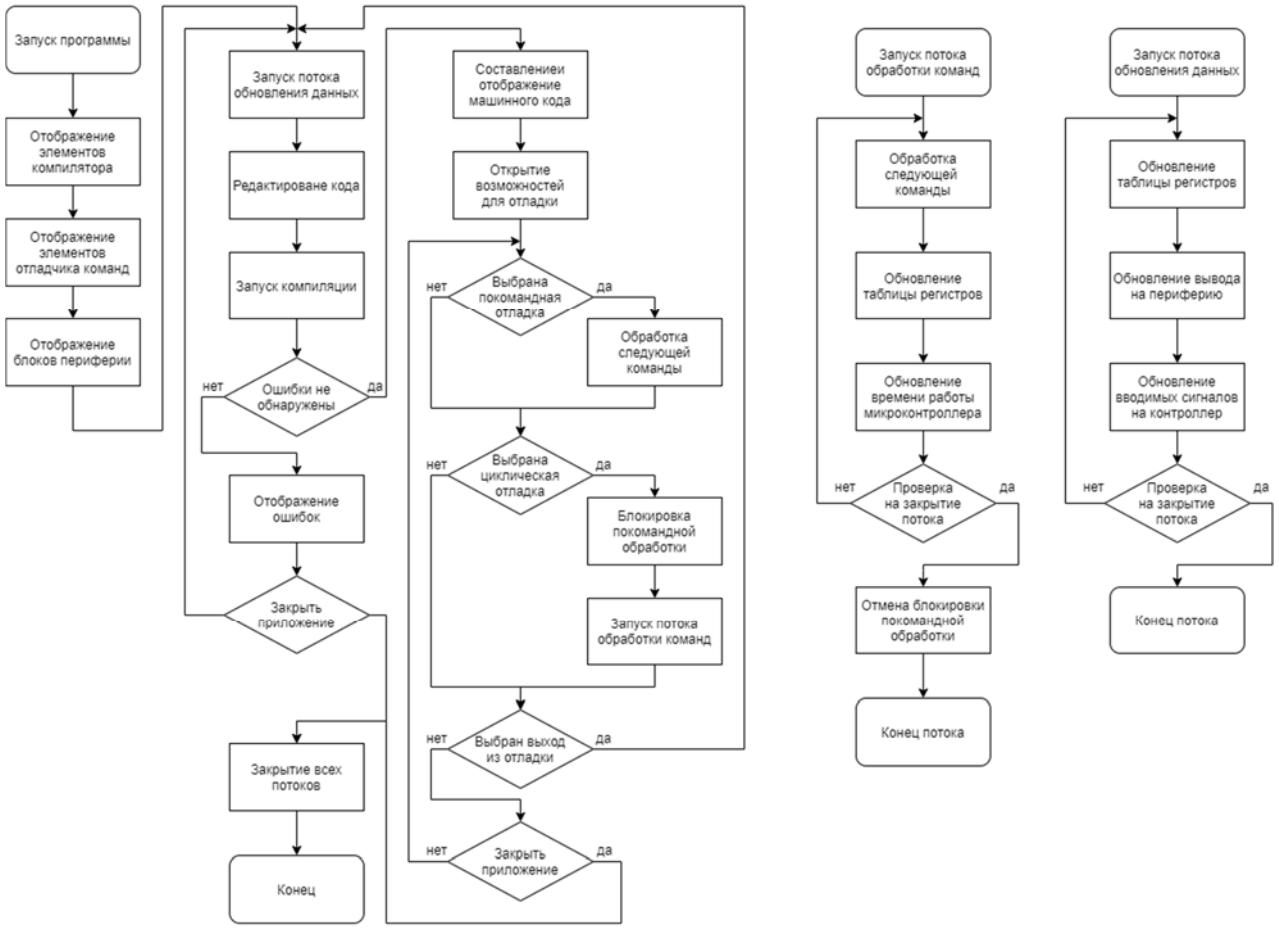


Рисунок 3.1 – Блок схема состояний

В реализованном приложении пользователь имеет большой набор инструментов для обучения программированию PIC микроконтроллеров и изучениях архитектуры данных микроконтроллеров. Для написания команды пользователь может использовать следующие возможности:

– редактировать кода программы;

– проверка кода на ошибки;

– просмотр ошибок;

– компиляция программы в машинный код;

– просмотр Program Memory микроконтроллера;

– просмотр стека микроконтроллера;

– просмотр регистров микроконтроллера;

– просмотр времени работы микроконтроллера;

– пошаговая обработка команд;

– циклическая обработка команд;

– расстановка точек остановки;

– удаление точек остановки;

– изменение задержки перед циклической обработкой команд;

– выбора порта для подключения светодиодов;

– просмотр цифрового вывода микроконтроллера;

– подключение кнопок к контактам микроконтроллера;

– цифровой ввод сигналов в память микроконтроллера.

Данного функционала уже достаточно для начала обучения начинающих специалистов, которые в бедующем смогут полноценно заняться программированием PIC микроконтроллеров или проектированием схожей техники. Однако на данном этапе разработок созданное приложение не в полной мере повторяет поведение микроконтроллера. На рисунке 3.2 отображена уже реализованная архитектура микроконтроллера в приложении. В первую очередь были выбраны именно эти блоки микроконтроллера PIC16F887, для реализации в программе. Если обратить внимании на предмет, связанный с данным контроллером в нашем учебном плане, то именно блоки мы изучали. И в данной программе можно с удобством запускать программы, которые мы составляли на данном предмете.

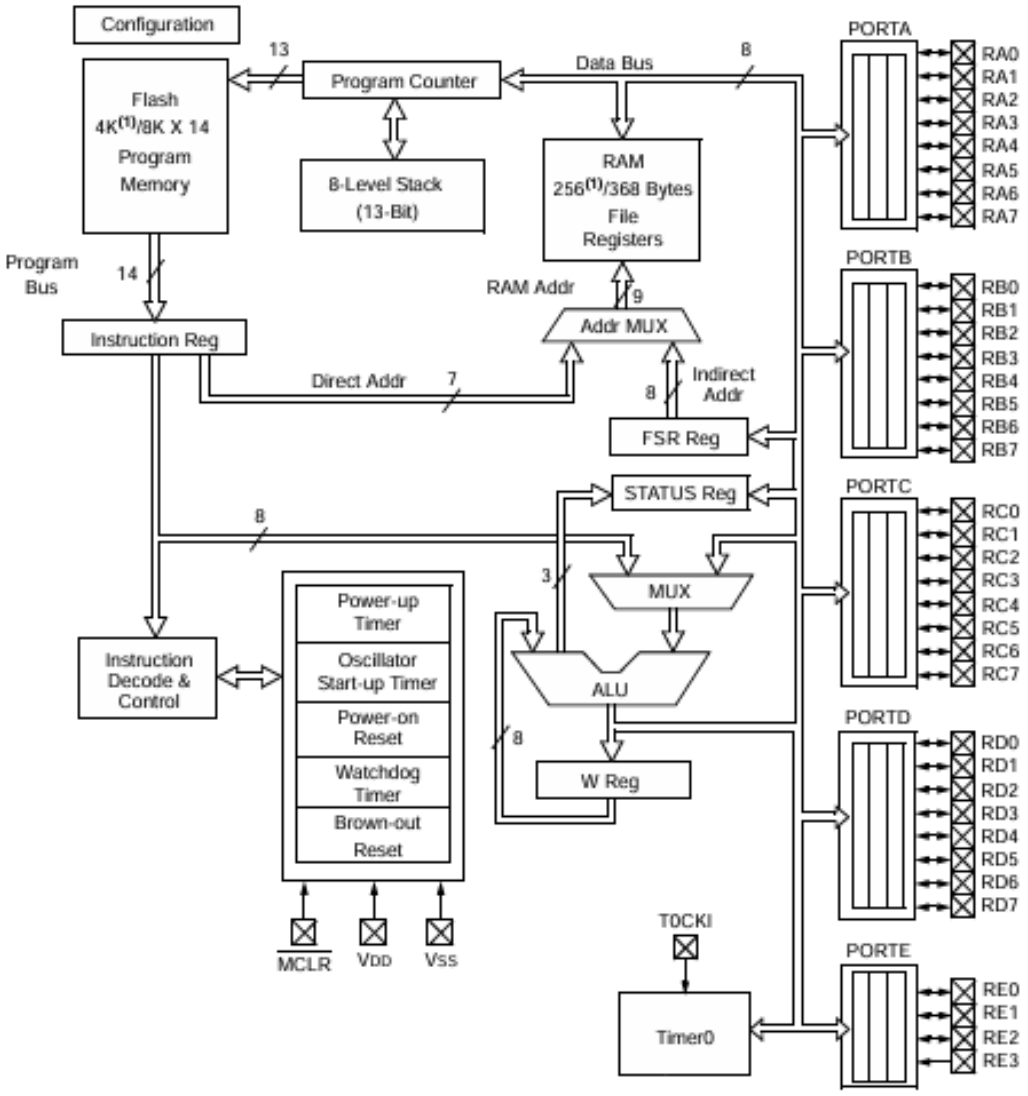


Рисунок 3.2 – Реализованная архитектура микроконтроллера

3.2 Рекомендации к дальнейшей разработке

На данный момент хоть приложение уже может представить необходимый функционал для обучения, но все же в контроллере осталась часть нереализованных регистров, отвечающих за работу периферии. Таким образом в дальнейшем необходимо реализовать следующие блоки микроконтроллера:

– Timer1;

– Timer2;

– Oscillator;

– Analog-To-Digital Converter;

– EUSART;

– 2 Analog Comparators and Reference;

– ECCP;

– CCP2;

– In-Circuit Debugger (ICD);

– Master Synchronous Serial Port (MSSP);

– 256 Bytes Data EEPROM.

Дальнейшая реализация программы непременно поможет в понимании о том, как работает вся периферия. После качественного обучения специалист с легкостью сможет подключить различную технику к PIC микроконтроллеру и управлять ей, не боясь за выход техники из строя.

Так же возможно добавление подсветки написанного кода в соответствии с машинном кодом, чтобы пользователь смог сопоставлять типы данных, так же можно реализовать меню выбора периферии, чтобы приложение являлось не статичным стендом, а полностью настраиваемым под нужды обучения.

4 Безопасность жизнедеятельности

4.1 Правила безопасности при использовании ПК

Правила выполнения работ с ПК обязаны обеспечивать минимизацию основных рисков, перечисленных в ГОСТ 12.0.003-2015. В первую очередь к ним относятся:

– высокий уровень напряженности электрического и магнитного полей в зоне установки компьютера;

– повышенная температура отдельных элементов персонального компьютера;

– монотонность трудового процесса;

– высокий уровень зрительных нагрузок;

– вероятность поражения статическим электричеством, генерируемым техникой;

– недостаточная освещенность в зоне выполнения операций у компьютера;

– другие факторы.

Полностью избавиться от их действия, конечно, невозможно, однако пользователь приложения обязан стремиться к тому, чтобы сократить воздействие этих факторов себя, обеспечивая возможность восстановления нормального самочувствия, работоспособности и обучаемости, не подвергая себя всевозможным рискам [8].

Заключение

При выполнение представленной работы был изучен объектно-ориентированный язык программирования C#, а также технология Windows Forms от компании Microsoft.

Был реализован компилятор с возможностью проверки написанного кода и возможностью компиляции программы в машинный код. Также реализована подсветка битов в командах машинного кода.

Был изучен реализованный PIC микроконтроллер, его типы данных, структура команд, работа портов ввода вывода, работа счетчика команд, работа прямой и косвенной адресаций, работа счетчика-таймера Timer0, регистров микроконтроллера, работа RC-генератора. Каждый из этих элементов был отображен в работе приложения симулятора PIC микроконтроллера.

В дальнейшем можно расширять функционал приложения, добавляя новые функции такие как подсветка команд в написанном коде, дополнительная периферия, более плавная настройка интерфейса программы. Более отзывчивый интерфейс при возникновении ошибок, возможность предлагать вводимые команды.

Список использованных источников

1. Прата Стивен. "Язык программирования С. Лекции и упражнения", 2018. - 928с.
2. learn.microsoft.com.: [Электронный ресурс]. *URL*: *https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/* (дата обращения 20.05.2025)
3. Брайан Нойес. " Привязка данных в Windows Forms", 2009. – 632с
4. qna.habr.com.: [Электронный ресурс]. *URL*: *https://qna.habr.com/q/487985* (дата обращения 11.05.2025)
5. Microchip Technology "PIC16F882/883/884/886/887 Data Sheet", 2006. - 328 с
6. metanit.com.: [Электронный ресурс]. *URL*: *https://metanit.com/sharp/tutorial/1.1.php* (дата обращения 22.05.2025)
7. ci-sharp.ru.: [Электронный ресурс]. *URL*: *https://ci-sharp.ru/obuchenie/oshibki-i-ih-resheniya/kak-podschitat-kolichestvo-vkhozhdenii-simvola-stroki-v-stroku-v-c/* (дата обращения 23.05.2025)
8. Информация о технике безопасности. [Электронный ресурс]. URL: *http s://www.centrattek.ru/info/tekhnika-bezopasnosti-pri-rabote-s-kompyuterom-razyasneniya/* (дата обращения 24.05.25)

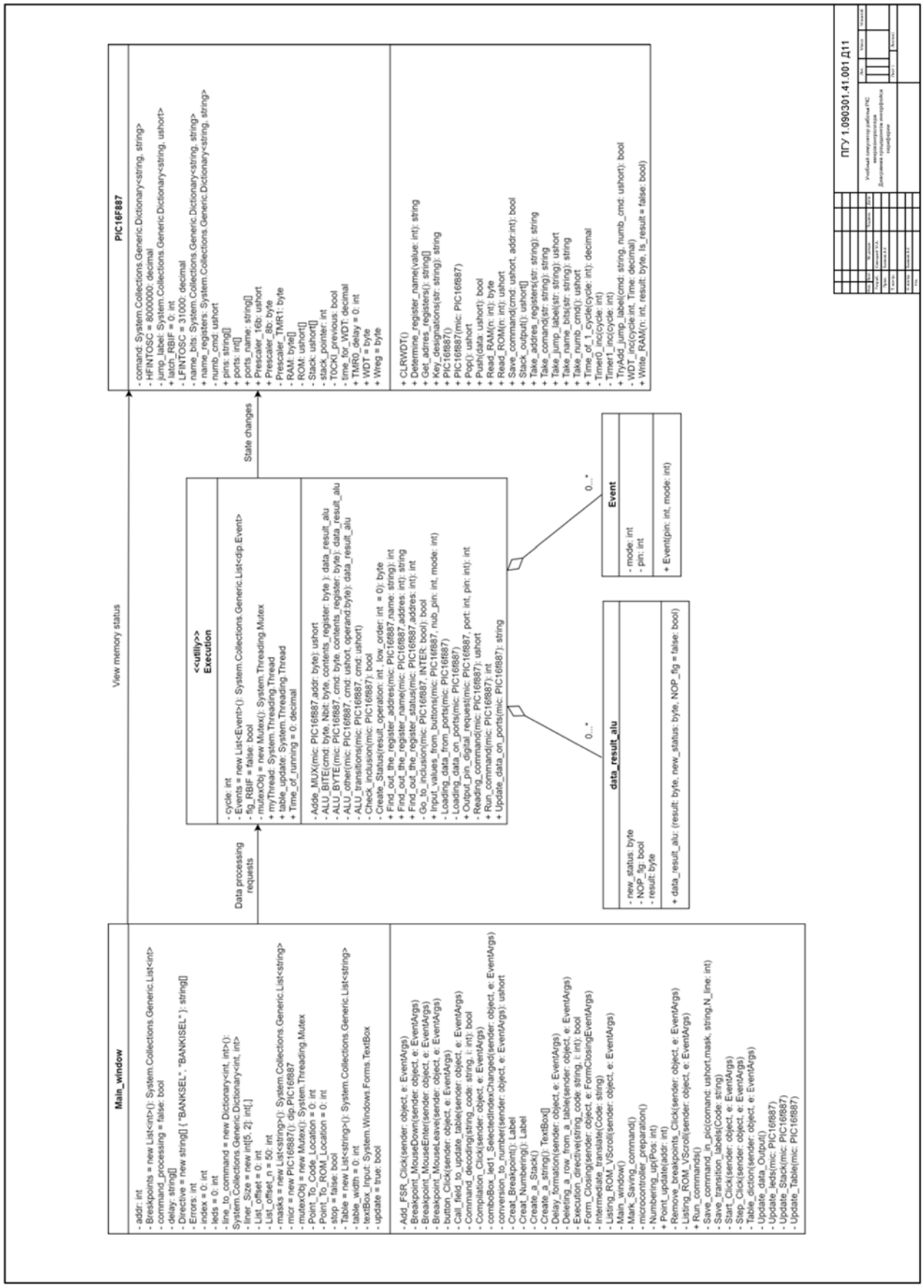
Приложение А

(Обязательное)

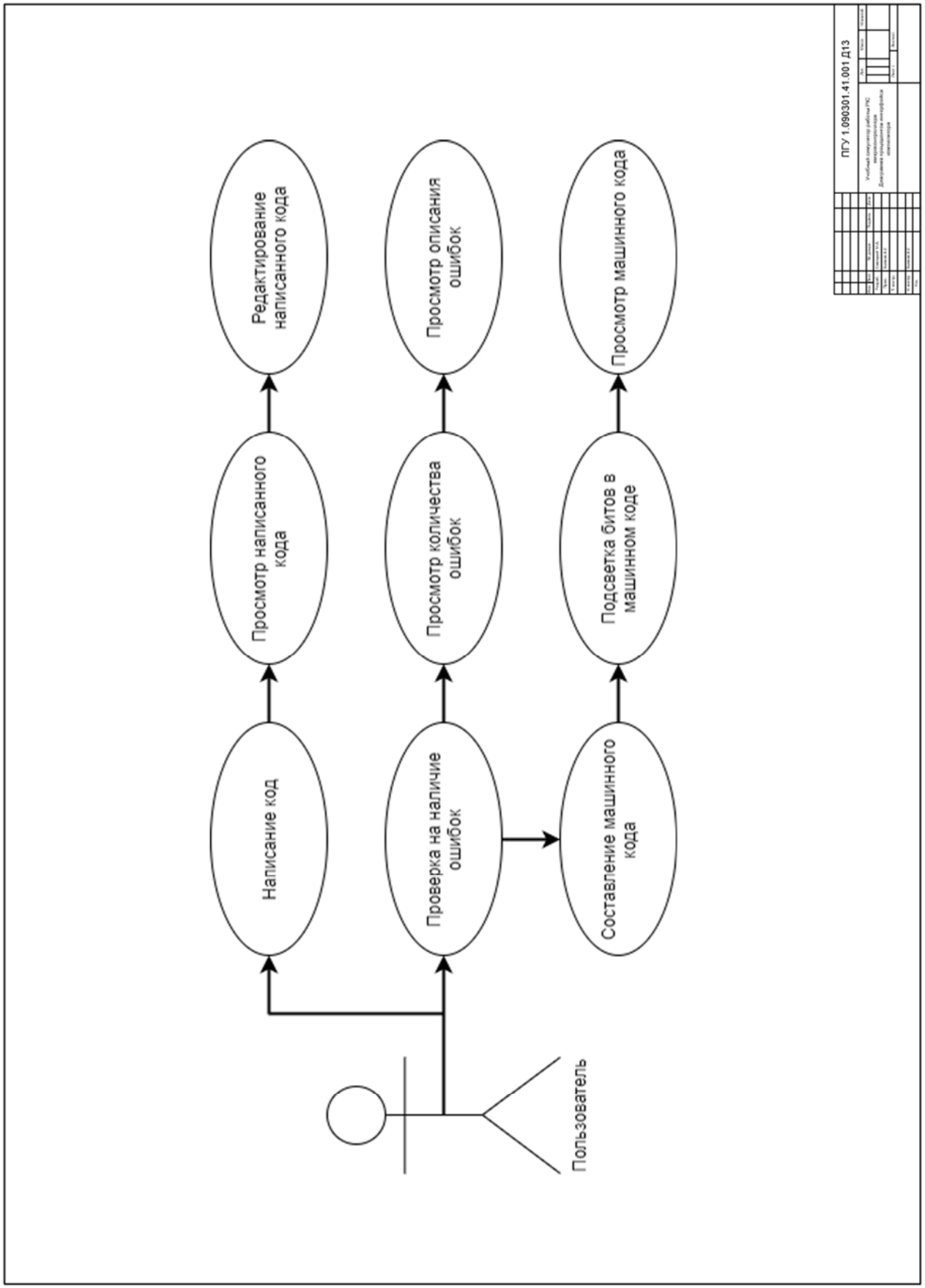
Учебный симулятор работы PIC микроконтроллера

UML диаграммы

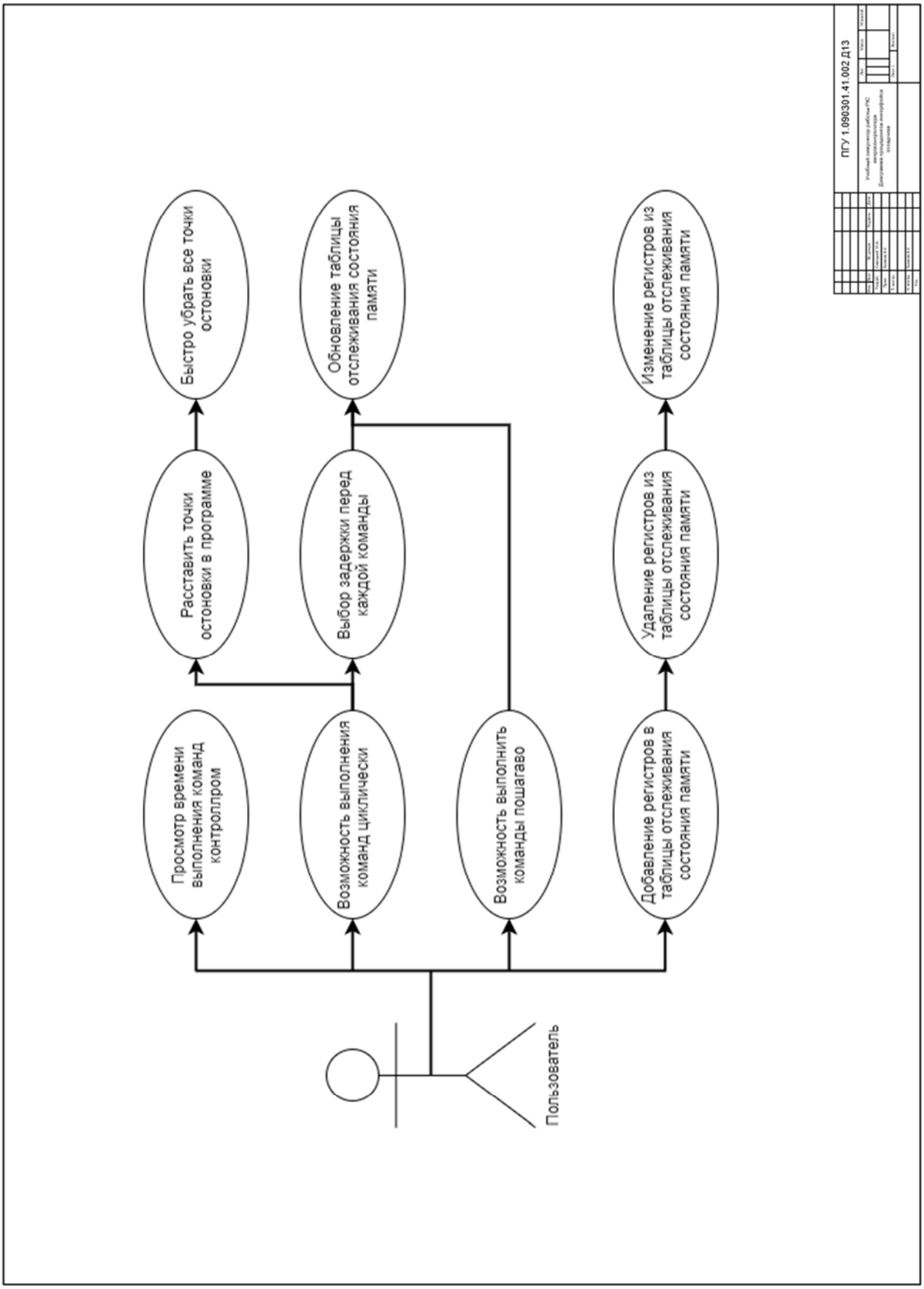
Приложение А.1 – *UML* – диаграмма классов



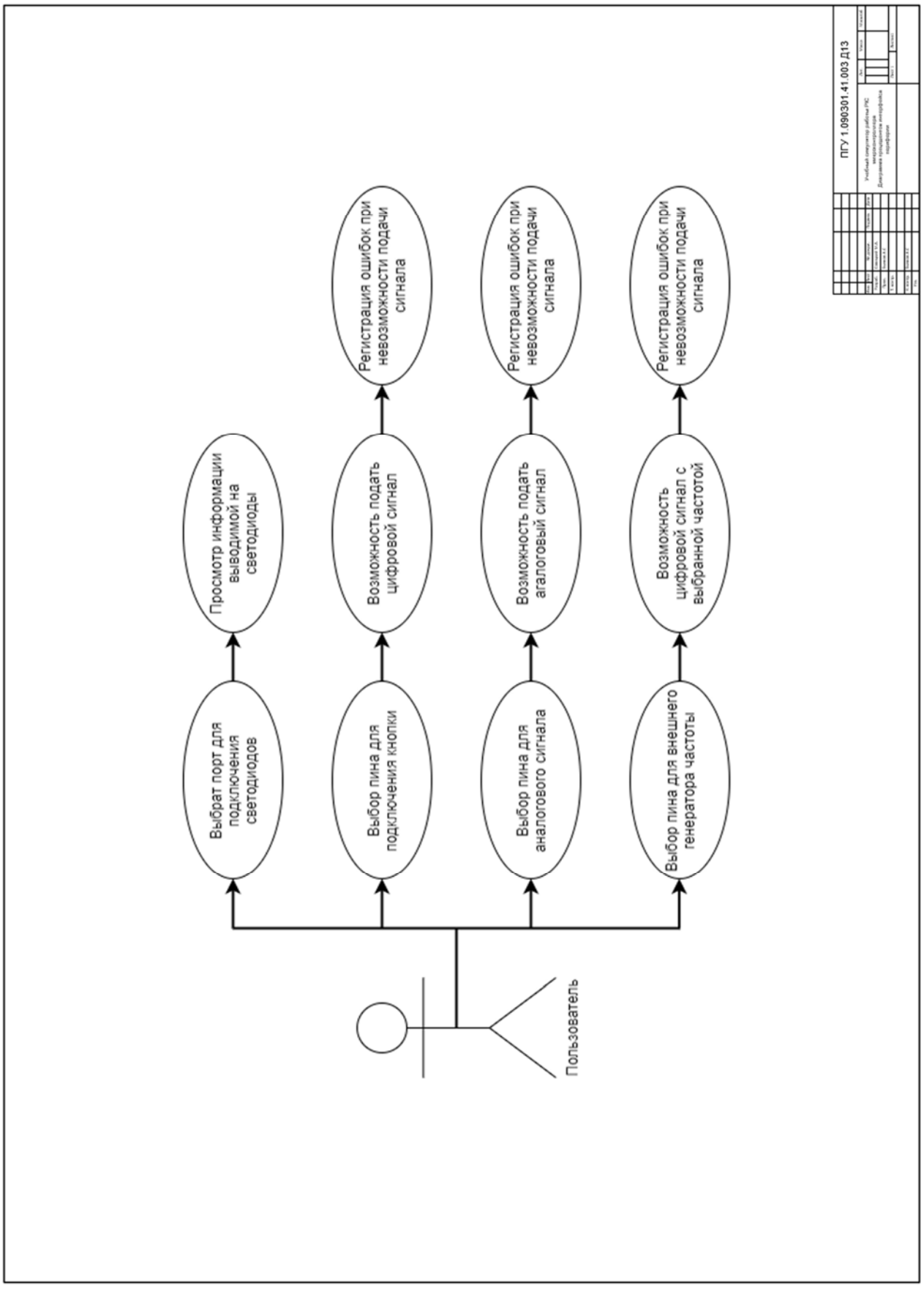
Приложение А.2 – *UML* – диаграмма прецедентов интерфейса компилятора



Приложение А.3 – *UML* – диаграмма прецедентов интерфейса отладчика



Приложение А.4 – *UML* – диаграмма прецедентов интерфейса периферии



Приложение Б

(Обязательное)

Учебный симулятор работы PIC микроконтроллера

Листинг программы

Файл Program.cs

```
namespace dip
{
    internal static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as
set high DPI settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Main_window());
        }
    }
}
```

Файл Main_window.Designer.cs

```
using System;
using System.Data.SqlTypes;
using System.Drawing;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace dip
{
    partial class Main_window
    {
        // настройка прокрутки Debug_console и Line_numbering
```

```
        [DllImport("user32.dll")]
        public static extern int GetScrollPos(IntPtr hwnd, int
nBar);
        private void Code_VScroll(object sender, EventArgs e)
        {
            int Pos;
            if (Code.Text.Count(c => c == '\n') + 1 >= 32)
                Pos = GetScrollPos(Code.Handle,
(int)System.Windows.Forms.Orientation.Vertical);
            else
                Pos = 0;
            Numbering_up(Pos);
        }

        /// <summary>
        ///  Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components =
null;

        /// <summary>
        ///  Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources
should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #region Windows Form Designer generated code

        /// <summary>
        ///  Required method for Designer support - do not
modify
        ///  the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            Label Aders;
            Label Register_name;
            Label Hex;
            Label Dec;
            Label Binari;
            Label Min_l;
            Label label2;
            Label Aders_Stack;
            Label label5;
            Compilation = new Button();
            Code = new RichTextBox();
            Start = new Button();
            Debug_console = new RichTextBox();
            Listing_ROM = new RichTextBox();
            comboBox_button1 = new ComboBox();
            comboBox_button2 = new ComboBox();
            comboBox_button4 = new ComboBox();
            comboBox_button3 = new ComboBox();
            button1 = new Button();
            button2 = new Button();
            button4 = new Button();
            button3 = new Button();
            comboBox_led1 = new ComboBox();
            led1 = new PictureBox();
            led2 = new PictureBox();
```

```
led3 = new PictureBox();
led4 = new PictureBox();
led8 = new PictureBox();
led7 = new PictureBox();
led6 = new PictureBox();
led5 = new PictureBox();
Register_display = new ComboBox();
Add_FSR = new Button();
groupBox1 = new GroupBox();
groupBox2 = new GroupBox();
View_RAM = new GroupBox();
Register_status = new FlowLayoutPanel();
trackBar1 = new TrackBar();
label1 = new Label();
Analog_input = new GroupBox();
comboBox1 = new ComboBox();
textBox1 = new TextBox();
textBox2 = new TextBox();
Time = new Label();
Step = new Button();
Delay = new TextBox();
Numbering = new Panel();
BreakpointsView = new Panel();
Pointer_To_Code = new Label();
Pointer_To_ROM = new Label();
Pointer_To_ROM_panel = new Panel();
Pointer_To_Code_panel = new Panel();
Remove_breakpoints = new Button();
groupBox3 = new GroupBox();
label4 = new Label();
label3 = new Label();
comboBox2 = new ComboBox();
button5 = new Button();
comboBox3 = new ComboBox();
```

```
            textBox3 = new TextBox();
            Stack_Output = new GroupBox();
            Stack_view = new FlowLayoutPanel();
            Aders = new Label();
            Register_name = new Label();
            Hex = new Label();
            Dec = new Label();
            Binari = new Label();
            Min_l = new Label();
            label2 = new Label();
            Aders_Stack = new Label();
            label5 = new Label();

((System.ComponentModel.ISupportInitialize)led1).BeginInit();

((System.ComponentModel.ISupportInitialize)led2).BeginInit();

((System.ComponentModel.ISupportInitialize)led3).BeginInit();

((System.ComponentModel.ISupportInitialize)led4).BeginInit();

((System.ComponentModel.ISupportInitialize)led8).BeginInit();

((System.ComponentModel.ISupportInitialize)led7).BeginInit();

((System.ComponentModel.ISupportInitialize)led6).BeginInit();

((System.ComponentModel.ISupportInitialize)led5).BeginInit();
            groupBox1.SuspendLayout();
            groupBox2.SuspendLayout();
            View_RAM.SuspendLayout();

((System.ComponentModel.ISupportInitialize)trackBar1).BeginInit(
);
```

```
            Analog_input.SuspendLayout();
            Pointer_To_ROM_panel.SuspendLayout();
            Pointer_To_Code_panel.SuspendLayout();
            groupBox3.SuspendLayout();
            Stack_Output.SuspendLayout();
            SuspendLayout();
            //
            // Aders
            //
            Aders.BackColor = SystemColors.ActiveBorder;
            Aders.BorderStyle = BorderStyle.Fixed3D;
            Aders.Location = new Point(6, 61);
            Aders.Margin = new Padding(0);
            Aders.Name = "Aders";
            Aders.Size = new Size(62, 27);
            Aders.TabIndex = 40;
            Aders.Text = "Aders";
            Aders.TextAlign = ContentAlignment.MiddleCenter;
            //
            // Register_name
            //
            Register_name.BackColor = SystemColors.ActiveBorder;
            Register_name.BorderStyle = BorderStyle.Fixed3D;
            Register_name.Location = new Point(68, 61);
            Register_name.Margin = new Padding(0);
            Register_name.Name = "Register_name";
            Register_name.Size = new Size(120, 27);
            Register_name.TabIndex = 42;
            Register_name.Text = "Register name";
            Register_name.TextAlign =
ContentAlignment.MiddleCenter;
            //
            // Hex
            //
```

```
Hex.BackColor = SystemColors.ActiveBorder;
Hex.BorderStyle = BorderStyle.Fixed3D;
Hex.Location = new Point(188, 61);
Hex.Margin = new Padding(0);
Hex.Name = "Hex";
Hex.Size = new Size(60, 27);
Hex.TabIndex = 43;
Hex.Text = "Hex";
Hex.TextAlign = ContentAlignment.MiddleCenter;
//
// Dec
//
Dec.BackColor = SystemColors.ActiveBorder;
Dec.BorderStyle = BorderStyle.Fixed3D;
Dec.Location = new Point(248, 61);
Dec.Margin = new Padding(0);
Dec.Name = "Dec";
Dec.Size = new Size(50, 27);
Dec.TabIndex = 44;
Dec.Text = "Dec";
Dec.TextAlign = ContentAlignment.MiddleCenter;
//
// Binari
//
Binari.BackColor = SystemColors.ActiveBorder;
Binari.BorderStyle = BorderStyle.Fixed3D;
Binari.Location = new Point(298, 61);
Binari.Margin = new Padding(0);
Binari.Name = "Binari";
Binari.Size = new Size(156, 27);
Binari.TabIndex = 45;
Binari.Text = "Binari";
Binari.TextAlign = ContentAlignment.MiddleCenter;
//
// Min_l
```

```
            //
            Min_l.Location = new Point(52, 70);
            Min_l.Name = "Min_l";
            Min_l.Size = new Size(43, 27);
            Min_l.TabIndex = 46;
            Min_l.Text = "Min:";
            Min_l.TextAlign = ContentAlignment.MiddleCenter;
            //
            // label2
            //
            label2.Location = new Point(52, 103);
            label2.Name = "label2";
            label2.Size = new Size(43, 27);
            label2.TabIndex = 47;
            label2.Text = "Max:";
            label2.TextAlign = ContentAlignment.MiddleCenter;
            //
            // Aders_Stack
            //
            Aders_Stack.BackColor = SystemColors.ActiveBorder;
            Aders_Stack.BorderStyle = BorderStyle.Fixed3D;
            Aders_Stack.Location = new Point(10, 37);
            Aders_Stack.Margin = new Padding(0);
            Aders_Stack.Name = "Aders_Stack";
            Aders_Stack.Size = new Size(39, 27);
            Aders_Stack.TabIndex = 41;
            Aders_Stack.Text = "Adr";
            Aders_Stack.TextAlign =
ContentAlignment.MiddleCenter;
            //
            // label5
            //
            label5.BackColor = SystemColors.ActiveBorder;
            label5.BorderStyle = BorderStyle.Fixed3D;
            label5.Location = new Point(49, 37);
```

```
label5.Margin = new Padding(0);
label5.Name = "label5";
label5.Size = new Size(101, 27);
label5.TabIndex = 46;
label5.Text = "Value";
label5.TextAlign = ContentAlignment.MiddleCenter;
//
// Compilation
//
Compilation.Location = new Point(285, 10);
Compilation.Name = "Compilation";
Compilation.Size = new Size(110, 30);
Compilation.TabIndex = 0;
Compilation.TabStop = false;
Compilation.Text = "Compilation";
Compilation.UseVisualStyleBackColor = true;
Compilation.Click += Compilation_Click;
//
// Code
//
Code.Location = new Point(111, 41);
Code.Margin = new Padding(0);
Code.Name = "Code";
Code.ScrollBars = RichTextBoxScrollBars.ForcedBoth;
Code.Size = new Size(553, 660);
Code.TabIndex = 2;
Code.TabStop = false;
Code.Text = "";
Code.WordWrap = false;
Code.VScroll += Code_VScroll;
Code.TextChanged += Code_VScroll;
//
// Start
//
```

```
            Start.Enabled = false;
            Start.Location = new Point(175, 10);
            Start.Name = "Start";
            Start.Size = new Size(110, 30);
            Start.TabIndex = 3;
            Start.TabStop = false;
            Start.Text = "Start";
            Start.UseVisualStyleBackColor = true;
            Start.Click += Start_Click;
            //
            // Debug_console
            //
            Debug_console.Location = new Point(12, 707);
            Debug_console.Name = "Debug_console";
            Debug_console.ReadOnly = true;
            Debug_console.Size = new Size(652, 126);
            Debug_console.TabIndex = 4;
            Debug_console.TabStop = false;
            Debug_console.Text = "";
            //
            // Listing_ROM
            //
            Listing_ROM.Location = new Point(695, 41);
            Listing_ROM.Name = "Listing_ROM";
            Listing_ROM.ReadOnly = true;
            Listing_ROM.ScrollBars =
RichTextBoxScrollBars.ForcedBoth;
            Listing_ROM.ShowSelectionMargin = true;
            Listing_ROM.Size = new Size(180, 792);
            Listing_ROM.TabIndex = 6;
            Listing_ROM.TabStop = false;
            Listing_ROM.Text = "";
            Listing_ROM.VScroll += Listing_ROM_VScroll;
            //
            // comboBox_button1
```

```
            //
            comboBox_button1.DropDownHeight = 200;
            comboBox_button1.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox_button1.DropDownWidth = 20;
            comboBox_button1.FormattingEnabled = true;
            comboBox_button1.IntegralHeight = false;
            comboBox_button1.Location = new Point(80, 36);
            comboBox_button1.Name = "comboBox_button1";
            comboBox_button1.Size = new Size(63, 28);
            comboBox_button1.TabIndex = 7;
            comboBox_button1.TabStop = false;
            //
            // comboBox_button2
            //
            comboBox_button2.DropDownHeight = 200;
            comboBox_button2.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox_button2.DropDownWidth = 20;
            comboBox_button2.FormattingEnabled = true;
            comboBox_button2.IntegralHeight = false;
            comboBox_button2.Location = new Point(80, 70);
            comboBox_button2.Name = "comboBox_button2";
            comboBox_button2.Size = new Size(63, 28);
            comboBox_button2.TabIndex = 12;
            comboBox_button2.TabStop = false;
            //
            // comboBox_button4
            //
            comboBox_button4.DropDownHeight = 200;
            comboBox_button4.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox_button4.DropDownWidth = 20;
            comboBox_button4.FormattingEnabled = true;
            comboBox_button4.IntegralHeight = false;
```

```
            comboBox_button4.Location = new Point(227, 70);
            comboBox_button4.Name = "comboBox_button4";
            comboBox_button4.Size = new Size(63, 28);
            comboBox_button4.TabIndex = 20;
            comboBox_button4.TabStop = false;
            //
            // comboBox_button3
            //
            comboBox_button3.DropDownHeight = 200;
            comboBox_button3.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox_button3.DropDownWidth = 20;
            comboBox_button3.FormattingEnabled = true;
            comboBox_button3.IntegralHeight = false;
            comboBox_button3.Location = new Point(227, 36);
            comboBox_button3.Name = "comboBox_button3";
            comboBox_button3.Size = new Size(63, 28);
            comboBox_button3.TabIndex = 15;
            comboBox_button3.TabStop = false;
            //
            // button1
            //
            button1.Location = new Point(6, 36);
            button1.Name = "button1";
            button1.Size = new Size(68, 28);
            button1.TabIndex = 8;
            button1.TabStop = false;
            button1.Text = "OFF";
            button1.UseVisualStyleBackColor = true;
            button1.Click += button_Click;
            //
            // button2
            //
            button2.Location = new Point(6, 70);
            button2.Name = "button2";
```

```
            button2.Size = new Size(68, 28);
            button2.TabIndex = 11;
            button2.TabStop = false;
            button2.Text = "OFF";
            button2.UseVisualStyleBackColor = true;
            button2.Click += button_Click;
            //
            // button4
            //
            button4.Location = new Point(153, 70);
            button4.Name = "button4";
            button4.Size = new Size(68, 28);
            button4.TabIndex = 19;
            button4.TabStop = false;
            button4.Text = "OFF";
            button4.UseVisualStyleBackColor = true;
            button4.Click += button_Click;
            //
            // button3
            //
            button3.Location = new Point(153, 36);
            button3.Name = "button3";
            button3.Size = new Size(68, 28);
            button3.TabIndex = 16;
            button3.TabStop = false;
            button3.Text = "OFF";
            button3.UseVisualStyleBackColor = true;
            button3.Click += button_Click;
            //
            // comboBox_led1
            //
            comboBox_led1.DropDownHeight = 200;
            comboBox_led1.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox_led1.DropDownWidth = 20;
```

```
            comboBox_led1.FormattingEnabled = true;
            comboBox_led1.IntegralHeight = false;
            comboBox_led1.Location = new Point(216, 31);
            comboBox_led1.Name = "comboBox_led1";
            comboBox_led1.Size = new Size(75, 28);
            comboBox_led1.TabIndex = 10;
            comboBox_led1.TabStop = false;
            comboBox_led1.SelectedIndexChanged +=
comboBox_led1_SelectedIndexChanged;
            //
            // led1
            //
            led1.BackColor = Color.Gray;
            led1.Location = new Point(190, 35);
            led1.Name = "led1";
            led1.Size = new Size(20, 20);
            led1.TabIndex = 9;
            led1.TabStop = false;
            //
            // led2
            //
            led2.BackColor = Color.Gray;
            led2.Location = new Point(164, 35);
            led2.Name = "led2";
            led2.Size = new Size(20, 20);
            led2.TabIndex = 13;
            led2.TabStop = false;
            //
            // led3
            //
            led3.BackColor = Color.Gray;
            led3.Location = new Point(138, 35);
            led3.Name = "led3";
            led3.Size = new Size(20, 20);
            led3.TabIndex = 17;
```

```
            led3.TabStop = false;
            //
            // led4
            //
            led4.BackColor = Color.Gray;
            led4.Location = new Point(112, 35);
            led4.Name = "led4";
            led4.Size = new Size(20, 20);
            led4.TabIndex = 21;
            led4.TabStop = false;
            //
            // led8
            //
            led8.BackColor = Color.Gray;
            led8.Location = new Point(8, 35);
            led8.Name = "led8";
            led8.Size = new Size(20, 20);
            led8.TabIndex = 29;
            led8.TabStop = false;
            //
            // led7
            //
            led7.BackColor = Color.Gray;
            led7.Location = new Point(34, 35);
            led7.Name = "led7";
            led7.Size = new Size(20, 20);
            led7.TabIndex = 27;
            led7.TabStop = false;
            //
            // led6
            //
            led6.BackColor = Color.Gray;
            led6.Location = new Point(60, 35);
            led6.Name = "led6";
            led6.Size = new Size(20, 20);
```

```
            led6.TabIndex = 25;
            led6.TabStop = false;
            //
            // led5
            //
            led5.BackColor = Color.Gray;
            led5.Location = new Point(86, 35);
            led5.Name = "led5";
            led5.Size = new Size(20, 20);
            led5.TabIndex = 23;
            led5.TabStop = false;
            //
            // Register_display
            //
            Register_display.DropDownHeight = 220;
            Register_display.DropDownStyle =
ComboBoxStyle.DropDownList;
            Register_display.FormattingEnabled = true;
            Register_display.IntegralHeight = false;
            Register_display.Location = new Point(6, 26);
            Register_display.Name = "Register_display";
            Register_display.Size = new Size(148, 28);
            Register_display.Sorted = true;
            Register_display.TabIndex = 32;
            Register_display.TabStop = false;
            //
            // Add_FSR
            //
            Add_FSR.Location = new Point(160, 25);
            Add_FSR.Name = "Add_FSR";
            Add_FSR.Size = new Size(94, 29);
            Add_FSR.TabIndex = 33;
            Add_FSR.TabStop = false;
            Add_FSR.Text = "Add FSR";
            Add_FSR.UseVisualStyleBackColor = true;
```

```
Add_FSR.Click += Add_FSR_Click;
//
// groupBox1
//
groupBox1.Controls.Add(led4);
groupBox1.Controls.Add(led3);
groupBox1.Controls.Add(led1);
groupBox1.Controls.Add(led6);
groupBox1.Controls.Add(comboBox_led1);
groupBox1.Controls.Add(led5);
groupBox1.Controls.Add(led7);
groupBox1.Controls.Add(led2);
groupBox1.Controls.Add(led8);
groupBox1.Location = new Point(881, 322);
groupBox1.Name = "groupBox1";
groupBox1.Size = new Size(298, 70);
groupBox1.TabIndex = 37;
groupBox1.TabStop = false;
groupBox1.Text = "LEDs";
//
// groupBox2
//
groupBox2.Controls.Add(button1);
groupBox2.Controls.Add(comboBox_button1);
groupBox2.Controls.Add(button3);
groupBox2.Controls.Add(comboBox_button3);
groupBox2.Controls.Add(button2);
groupBox2.Controls.Add(button4);
groupBox2.Controls.Add(comboBox_button4);
groupBox2.Controls.Add(comboBox_button2);
groupBox2.Location = new Point(881, 398);
groupBox2.Name = "groupBox2";
groupBox2.Size = new Size(298, 113);
groupBox2.TabIndex = 38;
groupBox2.TabStop = false;
```

```
            groupBox2.Text = "BUTTONs";
            //
            // View_RAM
            //
            View_RAM.Controls.Add(Aders);
            View_RAM.Controls.Add(Register_name);
            View_RAM.Controls.Add(Hex);
            View_RAM.Controls.Add(Dec);
            View_RAM.Controls.Add(Binari);
            View_RAM.Controls.Add(Add_FSR);
            View_RAM.Controls.Add(Register_status);
            View_RAM.Controls.Add(Register_display);
            View_RAM.Location = new Point(881, 41);
            View_RAM.Name = "View_RAM";
            View_RAM.Size = new Size(479, 275);
            View_RAM.TabIndex = 39;
            View_RAM.TabStop = false;
            View_RAM.Text = "View RAM";
            //
            // Register_status
            //
            Register_status.AutoScroll = true;
            Register_status.AutoSizeMode =
AutoSizeMode.GrowAndShrink;
            Register_status.BackColor = SystemColors.Window;
            Register_status.BorderStyle = BorderStyle.Fixed3D;
            Register_status.Location = new Point(6, 88);
            Register_status.Margin = new Padding(0);
            Register_status.Name = "Register_status";
            Register_status.Size = new Size(448, 180);
            Register_status.TabIndex = 41;
            //
            // trackBar1
            //
            trackBar1.Location = new Point(6, 26);
```

```
trackBar1.Name = "trackBar1";
trackBar1.Orientation = Orientation.Vertical;
trackBar1.Size = new Size(56, 144);
trackBar1.TabIndex = 40;
trackBar1.TabStop = false;
//
// label1
//
label1.Location = new Point(52, 142);
label1.Name = "label1";
label1.Size = new Size(92, 23);
label1.TabIndex = 41;
label1.Text = "label1";
label1.TextAlign = ContentAlignment.MiddleCenter;
//
// Analog_input
//
Analog_input.Controls.Add(label2);
Analog_input.Controls.Add(Min_l);
Analog_input.Controls.Add(comboBox1);
Analog_input.Controls.Add(textBox1);
Analog_input.Controls.Add(textBox2);
Analog_input.Controls.Add(trackBar1);
Analog_input.Controls.Add(label1);
Analog_input.Location = new Point(1153, 596);
Analog_input.Name = "Analog_input";
Analog_input.Size = new Size(152, 176);
Analog_input.TabIndex = 42;
Analog_input.TabStop = false;
Analog_input.Text = "Analog input";
Analog_input.Visible = false;
//
// comboBox1
//
```

```
comboBox1.FormattingEnabled = true;
comboBox1.Location = new Point(52, 36);
comboBox1.Name = "comboBox1";
comboBox1.Size = new Size(92, 28);
comboBox1.TabIndex = 45;
comboBox1.TabStop = false;
//
// textBox1
//
textBox1.Location = new Point(101, 70);
textBox1.Name = "textBox1";
textBox1.Size = new Size(43, 27);
textBox1.TabIndex = 43;
textBox1.TabStop = false;
//
// textBox2
//
textBox2.Location = new Point(101, 103);
textBox2.Name = "textBox2";
textBox2.Size = new Size(43, 27);
textBox2.TabIndex = 44;
textBox2.TabStop = false;
//
// Time
//
Time.Location = new Point(693, 9);
Time.Name = "Time";
Time.Size = new Size(132, 31);
Time.TabIndex = 43;
Time.Text = "Time: 0";
Time.TextAlign = ContentAlignment.MiddleLeft;
//
// Step
//
Step.Enabled = false;
```

```
            Step.Location = new Point(65, 10);
            Step.Name = "Step";
            Step.Size = new Size(110, 30);
            Step.TabIndex = 44;
            Step.TabStop = false;
            Step.Text = "Step";
            Step.UseVisualStyleBackColor = true;
            Step.Click += Step_Click;
            //
            // Delay
            //
            Delay.Location = new Point(831, 9);
            Delay.MaxLength = 5;
            Delay.Name = "Delay";
            Delay.Size = new Size(58, 27);
            Delay.TabIndex = 45;
            Delay.TabStop = false;
            Delay.Text = "200";
            Delay.Leave += Delay_formation;
            //
            // Numbering
            //
            Numbering.BackColor = SystemColors.Control;
            Numbering.Location = new Point(65, 43);
            Numbering.Name = "Numbering";
            Numbering.Size = new Size(45, 637);
            Numbering.TabIndex = 5;
            Numbering.Text = "1\n\n\n";
            //
            // BreakpointsView
            //
            BreakpointsView.BackColor = SystemColors.Window;
            BreakpointsView.BackgroundImageLayout =
ImageLayout.None;
            BreakpointsView.BorderStyle =
```

```
BorderStyle.FixedSingle;
            BreakpointsView.Location = new Point(39, 43);
            BreakpointsView.Name = "BreakpointsView";
            BreakpointsView.Size = new Size(24, 637);
            BreakpointsView.TabIndex = 52;
            //
            // Pointer_To_Code
            //
            Pointer_To_Code.AutoSize = true;
            Pointer_To_Code.BackColor = Color.Transparent;
            Pointer_To_Code.CausesValidation = false;
            Pointer_To_Code.Enabled = false;
            Pointer_To_Code.Location = new Point(0, 3);
            Pointer_To_Code.Name = "Pointer_To_Code";
            Pointer_To_Code.Size = new Size(24, 20);
            Pointer_To_Code.TabIndex = 54;
            Pointer_To_Code.Text = "➔";
            Pointer_To_Code.Visible = false;
            //
            // Pointer_To_ROM
            //
            Pointer_To_ROM.AutoSize = true;
            Pointer_To_ROM.Enabled = false;
            Pointer_To_ROM.Location = new Point(0, 0);
            Pointer_To_ROM.Name = "Pointer_To_ROM";
            Pointer_To_ROM.Size = new Size(24, 20);
            Pointer_To_ROM.TabIndex = 54;
            Pointer_To_ROM.Text = "➔";
            Pointer_To_ROM.Visible = false;
            //
            // Pointer_To_ROM_panel
            //
            Pointer_To_ROM_panel.BackColor =
SystemColors.Control;
```

```
            Pointer_To_ROM_panel.BackgroundImageLayout =
ImageLayout.None;
            Pointer_To_ROM_panel.Controls.Add(Pointer_To_ROM);
            Pointer_To_ROM_panel.Location = new Point(667, 41);
            Pointer_To_ROM_panel.Name = "Pointer_To_ROM_panel";
            Pointer_To_ROM_panel.Size = new Size(24, 792);
            Pointer_To_ROM_panel.TabIndex = 53;
            //
            // Pointer_To_Code_panel
            //
            Pointer_To_Code_panel.BackColor =
SystemColors.Control;
            Pointer_To_Code_panel.BackgroundImageLayout =
ImageLayout.None;
            Pointer_To_Code_panel.Controls.Add(Pointer_To_Code);
            Pointer_To_Code_panel.Location = new Point(12, 43);
            Pointer_To_Code_panel.Name =
"Pointer_To_Code_panel";
            Pointer_To_Code_panel.Size = new Size(24, 637);
            Pointer_To_Code_panel.TabIndex = 55;
            //
            // Remove_breakpoints
            //
            Remove_breakpoints.Location = new Point(395, 10);
            Remove_breakpoints.Name = "Remove_breakpoints";
            Remove_breakpoints.Size = new Size(110, 30);
            Remove_breakpoints.TabIndex = 56;
            Remove_breakpoints.TabStop = false;
            Remove_breakpoints.Text = "Remove bp";
            Remove_breakpoints.UseVisualStyleBackColor = true;
            Remove_breakpoints.Click +=
Remove_breakpoints_Click;
            //
            // groupBox3
            //
```

```
groupBox3.Controls.Add(label4);
groupBox3.Controls.Add(label3);
groupBox3.Controls.Add(comboBox2);
groupBox3.Controls.Add(button5);
groupBox3.Controls.Add(comboBox3);
groupBox3.Controls.Add(textBox3);
groupBox3.Location = new Point(915, 610);
groupBox3.Name = "groupBox3";
groupBox3.Size = new Size(204, 108);
groupBox3.TabIndex = 57;
groupBox3.TabStop = false;
groupBox3.Text = "groupBox3";
groupBox3.Visible = false;
//
// label4
//
label4.Location = new Point(8, 28);
label4.Name = "label4";
label4.Size = new Size(41, 27);
label4.TabIndex = 50;
label4.Text = "Pin";
label4.TextAlign = ContentAlignment.MiddleCenter;
//
// label3
//
label3.Location = new Point(8, 63);
label3.Name = "label3";
label3.Size = new Size(41, 27);
label3.TabIndex = 49;
label3.Text = "Freq.";
//
// comboBox2
//
comboBox2.DropDownHeight = 200;
comboBox2.DropDownStyle =
```

```
ComboBoxStyle.DropDownList;
            comboBox2.DropDownWidth = 20;
            comboBox2.FormattingEnabled = true;
            comboBox2.IntegralHeight = false;
            comboBox2.Location = new Point(127, 28);
            comboBox2.Name = "comboBox2";
            comboBox2.Size = new Size(63, 28);
            comboBox2.TabIndex = 48;
            comboBox2.TabStop = false;
            //
            // button5
            //
            button5.Location = new Point(56, 28);
            button5.Name = "button5";
            button5.Size = new Size(68, 29);
            button5.TabIndex = 47;
            button5.TabStop = false;
            button5.Text = "OFF";
            button5.UseVisualStyleBackColor = true;
            //
            // comboBox3
            //
            comboBox3.DropDownHeight = 200;
            comboBox3.DropDownStyle =
ComboBoxStyle.DropDownList;
            comboBox3.DropDownWidth = 20;
            comboBox3.FormattingEnabled = true;
            comboBox3.IntegralHeight = false;
            comboBox3.Location = new Point(127, 63);
            comboBox3.Name = "comboBox3";
            comboBox3.Size = new Size(63, 28);
            comboBox3.TabIndex = 46;
            comboBox3.TabStop = false;
            //
            // textBox3
```

```
            //
            textBox3.Location = new Point(58, 63);
            textBox3.Name = "textBox3";
            textBox3.Size = new Size(63, 27);
            textBox3.TabIndex = 44;
            textBox3.TabStop = false;
            //
            // Stack_Output
            //
            Stack_Output.Controls.Add(label5);
            Stack_Output.Controls.Add(Aders_Stack);
            Stack_Output.Controls.Add(Stack_view);
            Stack_Output.Location = new Point(1183, 322);
            Stack_Output.Name = "Stack_Output";
            Stack_Output.Size = new Size(177, 189);
            Stack_Output.TabIndex = 58;
            Stack_Output.TabStop = false;
            Stack_Output.Text = "Stack";
            //
            // Stack_view
            //
            Stack_view.AutoScroll = true;
            Stack_view.AutoSizeMode =
AutoSizeMode.GrowAndShrink;
            Stack_view.BackColor = SystemColors.Window;
            Stack_view.BackgroundImageLayout = ImageLayout.None;
            Stack_view.BorderStyle = BorderStyle.Fixed3D;
            Stack_view.Location = new Point(10, 67);
            Stack_view.Name = "Stack_view";
            Stack_view.Size = new Size(162, 116);
            Stack_view.TabIndex = 0;
            //
            // Main_window
            //
            AutoScaleDimensions = new SizeF(120F, 120F);
```

```
            AutoScaleMode = AutoScaleMode.Dpi;
            AutoScroll = true;
            ClientSize = new Size(1382, 853);
            Controls.Add(Stack_Output);
            Controls.Add(groupBox3);
            Controls.Add(Remove_breakpoints);
            Controls.Add(Pointer_To_Code_panel);
            Controls.Add(Pointer_To_ROM_panel);
            Controls.Add(BreakpointsView);
            Controls.Add(Numbering);
            Controls.Add(Delay);
            Controls.Add(Step);
            Controls.Add(Time);
            Controls.Add(Analog_input);
            Controls.Add(View_RAM);
            Controls.Add(groupBox2);
            Controls.Add(groupBox1);
            Controls.Add(Listing_ROM);
            Controls.Add(Debug_console);
            Controls.Add(Start);
            Controls.Add(Code);
            Controls.Add(Compilation);
            Name = "Main_window";
            Text = "PIC16F887";
            FormClosing += Form_Closing;

((System.ComponentModel.ISupportInitialize)led1).EndInit();

((System.ComponentModel.ISupportInitialize)led2).EndInit();

((System.ComponentModel.ISupportInitialize)led3).EndInit();

((System.ComponentModel.ISupportInitialize)led4).EndInit();
```

```
((System.ComponentModel.ISupportInitialize)led8).EndInit();

((System.ComponentModel.ISupportInitialize)led7).EndInit();

((System.ComponentModel.ISupportInitialize)led6).EndInit();

((System.ComponentModel.ISupportInitialize)led5).EndInit();
            groupBox1.ResumeLayout(false);
            groupBox2.ResumeLayout(false);
            View_RAM.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)trackBar1).EndInit();
            Analog_input.ResumeLayout(false);
            Analog_input.PerformLayout();
            Pointer_To_ROM_panel.ResumeLayout(false);
            Pointer_To_ROM_panel.PerformLayout();
            Pointer_To_Code_panel.ResumeLayout(false);
            Pointer_To_Code_panel.PerformLayout();
            groupBox3.ResumeLayout(false);
            groupBox3.PerformLayout();
            Stack_Output.ResumeLayout(false);
            ResumeLayout(false);
            PerformLayout();
        }

        #endregion

        private Button Compilation;
        private RichTextBox Code;
        private Button Start;
        private RichTextBox Debug_console;
        private RichTextBox Listing_ROM;
        private ComboBox comboBox_button1;
        private Button button1;
```

```
private PictureBox led1;
private ComboBox comboBox_led1;
private Button button2;
private PictureBox led2;
private ComboBox comboBox_button2;
private PictureBox led4;
private ComboBox comboBox_button4;
private Button button4;
private PictureBox led3;
private Button button3;
private ComboBox comboBox_button3;
private PictureBox led8;
private PictureBox led7;
private PictureBox led6;
private PictureBox led5;
private ComboBox Register_display;
private Button Add_FSR;
private GroupBox groupBox1;
private GroupBox groupBox2;
private GroupBox View_RAM;
private FlowLayoutPanel Register_status;
private TrackBar trackBar1;
private Label label1;
private GroupBox Analog_input;
private ComboBox comboBox1;
private TextBox textBox1;
private TextBox textBox2;
private Label Time;
private Button Step;
private TextBox Delay;
private Panel Numbering;
private Panel BreakpointsView;
private Label Pointer_To_ROM;
private Panel Pointer_To_ROM_panel;
```

```
        private Panel Pointer_To_Code_panel;
        private Label Pointer_To_Code;
        private Button Remove_breakpoints;
        private GroupBox groupBox3;
        private Label label3;
        private ComboBox comboBox2;
        private Button button5;
        private ComboBox comboBox3;
        private TextBox textBox3;
        private Label label4;
        private GroupBox Stack_Output;
        private FlowLayoutPanel Stack_view;
    }
}
```

Файл Main_window.cs

```
using System;
using System.Reflection;
using System.Reflection.Metadata;
using System.Runtime.InteropServices;
using System.Runtime.Intrinsics.Arm;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using static System.Net.Mime.MediaTypeNames;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.Window;
using System.Threading;
using System.Runtime.CompilerServices;
using Microsoft.VisualBasic;
using static System.Collections.Specialized.BitVector32;
using System.Data.SqlTypes;
using static dip.Main_window;
using System.Drawing;
```

```
using System.Linq;
using System.Runtime.Serialization.Json;
using System.Threading.Tasks;
using System.Windows.Input;

namespace dip
{
    public partial class Main_window : Form
    {
        PIC16f887 micr = new PIC16f887();
        string[] Directive = new string[] { "BANKSEL",
"BANKISEL" };
        bool stop = false, command_processing = false, update
= true;
        int Errors;
        int leds = 0;
        int delay;
        int index = 0;
        int table_width = 0;
        int addr;
        int List_offset = 0;
        internal int List_offset_n = 50;
        int Point_To_Code_Location = 0;
        int Point_To_ROM_Location = 0;
        List<int> Breskpoints = new List<int>();
        Dictionary<int, int> line_to_command = new
Dictionary<int, int>();
        List<string> masks = new List<string>();
        TextBox? textBox_Input;
        Mutex mutexObj = new Mutex();
        List<string> Table = new List<string>();
        int[,] liner_Size = new int[5, 2];

        public Main_window()
        {
```

```
            InitializeComponent();
            comboBox_button1.Items.Add("non");
            comboBox_button1.Items.AddRange(micr.pins);
            comboBox_button1.SelectedIndex = 0;
            comboBox_button2.Items.Add("non");
            comboBox_button2.Items.AddRange(micr.pins);
            comboBox_button2.SelectedIndex = 0;
            comboBox_button3.Items.Add("non");
            comboBox_button3.Items.AddRange(micr.pins);
            comboBox_button3.SelectedIndex = 0;
            comboBox_button4.Items.Add("non");
            comboBox_button4.Items.AddRange(micr.pins);
            comboBox_button4.SelectedIndex = 0;

            comboBox_led1.Items.Add("non");
            comboBox_led1.Items.AddRange(micr.ports_name);
            comboBox_led1.SelectedIndex = 0;


Register_display.Items.AddRange(micr.Get_adrres_registers());
            Table.Add("");
            Register_display.SelectedIndex = 0;

Register_status.Controls.AddRange(Create_a_string());
            delay = Convert.ToInt32(Delay.Text);
            Listing_ROM.SelectionAlignment =
HorizontalAlignment.Right;

            Execution.table_update = new
Thread(Update_data_Output);
            Execution.table_update.Start();

            BreakpointsView.Controls.Add(Creat_Breakpoint());
            for (int i = 0; i < 64; i++)
            {
```

```
                Label L = Creat_Breakpoint();
                Numbering.Controls.Add(Creat_Numbering());
                BreakpointsView.Controls.Add(L);
            }
            Numbering_up(0);
            Create_a_Stack();
        }
        //копмиляция
        private ushort conversion_to_number(string
string_with_number, int number_bits)
        {
            if (number_bits <= 11 && number_bits > 0)
            {
                string_with_number = new
Regex(@"\s").Replace(string_with_number, "");

                if (micr.Key_designations(string_with_number)
== "jump_label")
                    return
(ushort)(micr.Take_jump_label(string_with_number) % (1 <<
number_bits));

                string_with_number =
string_with_number.ToUpper();

                //.123 | D'123'
                if (Regex.IsMatch(string_with_number, "^\\.[0-
9]+$") || Regex.IsMatch(string_with_number, "^D'[0-9]+'$"))
                {
                    string expected_translation = new
Regex(@"\D").Replace(string_with_number, "").PadLeft(18, '0');
                    return
(ushort)(Convert.ToInt64(expected_translation.Substring(expect
ed_translation.Length - 18)) % (1 << number_bits));
```

```
                }

                //B'011011'
                if (Regex.IsMatch(string_with_number, "^B'[0-
1]+'$"))
                {
                    string expected_translation = new
Regex(@"\D").Replace(string_with_number, "").PadLeft(11, '0');
                    return
(ushort)(Convert.ToInt32(expected_translation.Substring(expect
ed_translation.Length - number_bits), 2) % (1 <<
number_bits));
                }

                //O'17'
                if (Regex.IsMatch(string_with_number, "^O'[0-
7]+'$"))
                {
                    string expected_translation = new
Regex(@"\D").Replace(string_with_number, "").PadLeft(4, '0');
                    return
(ushort)(Convert.ToInt32(expected_translation.Substring(expect
ed_translation.Length - 4), 8) % (1 << number_bits));
                }

                //0X1A | 1A | H'1A'
                if (Regex.IsMatch(string_with_number,
"^0(X|x)([0-9]|A|B|C|D|E|F)+$") ||
Regex.IsMatch(string_with_number, "^([0-9]|A|B|C|D|E|F)+$") ||
Regex.IsMatch(string_with_number, "^H'([0-
9]|A|B|C|D|E|F)+'$"))
                {
                    string expected_translation = new
Regex(@"(?=\D)[^ABCDEF]").Replace(string_with_number,
"").PadLeft(3, '0');
```

```
                    return
(ushort)(Convert.ToInt32(expected_translation.Substring(expect
ed_translation.Length - 3), 16) % (1 << number_bits));
                }

                if (micr.Key_designations(string_with_number)
== "name_registers")
                {
                    string expected_translation =
Convert.ToString(Convert.ToInt32(micr.Take_addres_registers(st
ring_with_number), 16), 2).PadLeft(11, '0');
                    return
(ushort)(Convert.ToUInt16(expected_translation, 2) % (1 <<
number_bits));
                }
                if (micr.Key_designations(string_with_number)
== "name_bits")
                    return
(ushort)(Convert.ToUInt16(micr.Take_name_bits(string_with_numb
er), 2) % (1 << number_bits));

            }
            return 16384;
        }
        private bool Command_decoding(string string_code, int
i)
        {
            MatchCollection Matched = new
Regex(@"\S+").Matches(string_code);
            if (Matched.Count == 0)
                return false;

            if
(!(micr.Key_designations(Matched[0].Value.ToUpper()) ==
"comand"))
```

```
                return false;

            //если это все же команда
            string additional_data =
string_code.Substring(string_code.IndexOf(Matched[0].Value) +
Matched[0].Value.Length);

            string cmd =
micr.Take_comand(Matched[0].Value.ToUpper());
            cmd = cmd.Replace("x", "0");
            ushort comand = 0;
            int error = 0;
            if (Regex.IsMatch(cmd, "^[0-1]{14}$") &&
Regex.IsMatch(additional_data, "^\\s*$"))
            {
                Save_command_in_pic(Convert.ToUInt16(cmd, 2),
cmd, i);
                return true;
            }
            if (cmd.IndexOf("k") != -1 &&
Regex.IsMatch(additional_data, "^\\s*\\S[^,]+\\s*$"))
            {
                int number_bits = cmd.Length -
cmd.Replace("k", "").Length;
                ushort first_operand =
conversion_to_number(additional_data, number_bits);
                if (first_operand >= Math.Pow(2, number_bits))
                    error = 1;
                if (error == 0)
                    comand =
Convert.ToUInt16(Convert.ToUInt16(cmd.Replace("k", "0"), 2) |
first_operand);
            }
```

```
            if (cmd.IndexOf("f") != -1 && cmd.IndexOf("d") ==
-1 && cmd.IndexOf("b") == -1 && Regex.IsMatch(additional_data,
"^\\s*\\S[^,]+\\s*$"))
            {
                ushort first_operand =
conversion_to_number(additional_data, 7);
                if (first_operand >= Math.Pow(2, 7))
                    error = 1;
                if (error == 0)
                    comand =
Convert.ToUInt16(Convert.ToUInt16(cmd.Replace("f", "0"), 2) |
first_operand);
            }
            if (cmd.IndexOf("f") != -1 && cmd.IndexOf("b") !=
-1 && Regex.IsMatch(additional_data, "^\\s*\\S[^,]+\\s*$"))
            {
                ushort first_operand =
conversion_to_number(additional_data, 7);
                if (first_operand >= Math.Pow(2, 7))
                    error = 1;
                if (error == 0)
                    comand =
Convert.ToUInt16(Convert.ToUInt16(new
Regex(@"(f|b)").Replace(cmd, "0"), 2) | first_operand | 128);
            }

            if (cmd.IndexOf("f") != -1 && cmd.IndexOf("b") !=
-1 && Regex.IsMatch(additional_data,
"^\\s*\\S+\\s*,\\s*\\S+\\s*$"))
            {
                string[] part_string =
additional_data.Split(",");
                ushort first_operand =
conversion_to_number(part_string[0], 7);
                ushort second_operand =
```

```
conversion_to_number(part_string[1], 3);
                if (first_operand >= Math.Pow(2, 7))
                    error = (ushort)(error | 1);
                if (second_operand >= Math.Pow(2, 3))
                    error = (ushort)(error | 2);
                if (error == 0)
                    comand =
Convert.ToUInt16(Convert.ToUInt16(new
Regex(@"(f|b)").Replace(cmd, "0"), 2) | first_operand |
second_operand << 7);
            }

            if (cmd.IndexOf("f") != -1 && cmd.IndexOf("d") !=
-1 && Regex.IsMatch(additional_data,
"^\\s*\\S+\\s*,\\s*\\S+\\s*$"))
            {
                string[] part_string =
additional_data.Split(",");
                ushort first_operand =
conversion_to_number(part_string[0], 7);
                ushort second_operand =
conversion_to_number(part_string[1], 1);
                if (first_operand >= Math.Pow(2, 7))
                    error = (ushort)(error | 1);
                if (second_operand >= Math.Pow(2, 1))
                    error = (ushort)(error | 2);
                if (error == 0)
                    comand =
Convert.ToUInt16(Convert.ToUInt16(new
Regex(@"(f|d)").Replace(cmd, "0"), 2) | first_operand |
second_operand << 7);
            }
            if (!Regex.IsMatch(cmd, "^[0-1]{14}$") && error ==
0 && comand == 0)
            {
```

```
                error = 4;
            }
            switch (error)
            {
                case 1:
                    Errors++;
                    Debug_console.Text += "Ошибка (" + Errors +
") 'Некорректный ввод значения первого аргумента'  в " + (i + 1) + "
строке\n";
                    break;
                case 2:
                    Errors++;
                    Debug_console.Text += "Ошибка (" + Errors +
") 'Некорректный ввод значения второго аргумента'  в " + (i + 1) + "
строке\n";
                    break;
                case 3:
                    Errors++;
                    Debug_console.Text += "Ошибка (" + Errors +
") 'Некорректный ввод значений аргументов'  в " + (i + 1) + "
строке\n";
                    break;
                case 4:
                    Errors++;
                    Debug_console.Text += "Ошибка (" + Errors +
") 'Недостаточно аргументов в команде'  в " + (i + 1) + " строке\n";
                    break;
                default:
                    Save_command_in_pic(comand, cmd, i);
                    break;

            }
            return true;
        }
```

```
        private void Save_command_in_pic(ushort comand, string
mask, int N_line)
        {
            Listing_ROM.Text += addr + ": " +
Convert.ToString(comand, 2).PadLeft(14, '0') + "\n";
            if (N_line >= 0)
                line_to_command.Add(addr, N_line + 1);
            masks.Add(mask);
            micr.Save_command(comand, addr++);

        }
        private void Mark_Saving_command()
        {
            int addr_point = 0;
            string[] strs = Listing_ROM.Text.Split("\n");
            for (int i = 0; i < strs.Length - 1; i++)
            {
                addr_point += strs[i].IndexOf(":") + 2;
                for (int j = 0; j < 14; j++)
                {
                    Listing_ROM.SelectionStart = addr_point;
                    Listing_ROM.SelectionLength = 1;
                    switch ((masks[i])[j])
                    {
                        case '0':
                        case '1':
                            Listing_ROM.SelectionColor =
Color.DimGray;
                            break;
                        case 'b':
                            Listing_ROM.SelectionColor =
Color.DarkRed;
                            break;
                        case 'k':
                            Listing_ROM.SelectionColor =
```

```
Color.DarkCyan;
                            break;
                        case 'f':
                            Listing_ROM.SelectionColor =
Color.Purple;
                            break;
                        case 'd':
                            Listing_ROM.SelectionColor =
Color.Blue;
                            break;

                    }
                    addr_point++;
                }

                addr_point++;
            }
        }
        private void Intermediate_translate(string Code)
        {
            if (Code == "")
            {
                Errors++;
                Debug_console.Text += "Ошибка(" + Errors + ")
'Отсутсвует исполняемый код'\n";

                return;
            }
            string[] Code_strs = Code.Split("\n");
            for (int i = 0; i < Code_strs.Length; i++)
            {
                string string_code = Code_strs[i];
                if (string_code.IndexOf(";") > 0)
                    string_code = string_code.Substring(0,
```

```
string_code.IndexOf(";")); //убрать комментарий

                if (Regex.IsMatch(string_code, "^\\s*$"))
//пустая строка
                    continue;

                //убрать метку
                MatchCollection Matched = new
Regex(@"\S+").Matches(Code_strs[i]);
                if (micr.Key_designations(Matched[0].Value) ==
"jump_label")
                    string_code =
string_code.Substring(string_code.IndexOf(Matched[0].Value) +
Matched[0].Value.Length);

                if (Regex.IsMatch(string_code, "^\\s*$"))
//пустая строка
                    continue;

                if (Command_decoding(string_code, i)) //если это
команда
                    continue;

                if (Execution_directive(string_code, i)) //если
это деректива
                    continue;

                Errors++;
                Debug_console.Text += "Ошибка(" + Errors + ")
'Неизвестная команда'\n";
            }
        }
        private bool Execution_directive(string string_code,
int i)
```

```
        {
            MatchCollection Matched = new
Regex(@"\S+").Matches(string_code);
            if (Matched.Count == 0)
                return false;

            string additional_data =
string_code.Substring(string_code.IndexOf(Matched[0].Value) +
Matched[0].Value.Length);
            switch (Matched[0].Value.ToUpper())
            {
                case "BANKSEL":
                    if (Regex.IsMatch(additional_data,
"^\\s*\\S[^,]+\\s*$"))
                    {
                        int addr_registr =
conversion_to_number(additional_data, 9);
                        int RP0 = (addr_registr >> 7) & 1;
                        int RP1 = (addr_registr >> 8) & 1;
                        Save_command_in_pic((ushort)(4739 |
(RP0 << 10)), "0100bbbfffffff", i);
                        Save_command_in_pic((ushort)(4867 |
(RP1 << 10)), "0100bbbfffffff", -1);
                    }
                    else
                    {
                        Errors++;
                        Debug_console.Text += "Ошибка (" +
Errors + ") 'Неверное указание регистра'  в " + (i + 1) + "
строке\n";
                    }
                    return true;
                case "BANKISEL":
                    if (Regex.IsMatch(additional_data,
```

```
"^\\s*\\S[^,]+\\s*$"))
                    {
                        int IRP =
conversion_to_number(additional_data, 9) >> 9;
                        Listing_ROM.Text += addr + ": " +
Convert.ToString(4995 | (IRP << 10), 2).PadLeft(14, '0') +
"\n";
                        Save_command_in_pic((ushort)(4995 |
(IRP << 10)), "0100bbbfffffff", i);
                    }
                    else
                    {
                        Errors++;
                        Debug_console.Text += "Ошибка (" +
Errors + ") 'Неверное указание регистра'  в " + (i + 1) + "
строке\n";
                    }
                    return true;
                default:
                    return false;
            }
        }
        private void Save_transition_labels(string Code)
        {
            string[] Code_strs = Code.Split("\n");
            ushort numb_cmd = 0;
            for (int i = 0; i < Code_strs.Length; i++)
            {
                if (Code_strs[i].IndexOf(";") > 0)
                    Code_strs[i] = Code_strs[i].Substring(0,
Code_strs[i].IndexOf(";"));
                if (Regex.IsMatch(Code_strs[i], "^\\s*$"))
                    continue;
```

```
                MatchCollection Matched = new
Regex(@"\S+").Matches(Code_strs[i]);
                if (Regex.IsMatch(Matched[0].Value,
"^\\d\\w*$"))
                {
                    Errors++;
                    Debug_console.Text += "Ошибка (" + Errors +
") 'Команда не может начинатсья с цыфры' в " + (i + 1) + "
строке\n";
                    continue;
                }
                if
(micr.Key_designations(Matched[0].Value.ToUpper()) == "absent"
&& Array.IndexOf(Directive, Matched[0].Value.ToUpper()) == -1)
                {
                    if
(!micr.TryAdd_jump_label(Matched[0].Value, numb_cmd))
                    {
                        Errors++;
                        Debug_console.Text += "Ошибка (" +
Errors + ")'Дублирующая метка' в " + (i + 1) + " строке\n";
                    }
                    continue;
                }
                numb_cmd++;
                if (Array.IndexOf(Directive,
Matched[0].Value.ToUpper()) == 0)
                    numb_cmd++;
            }
        }
        private void microcontroller_preparation()
        {
            Errors = 0;
            addr = 0;
```

```
            masks.Clear();
            line_to_command.Clear();
            Debug_console.Text = "Компиляция началась\n";
            micr = new PIC16f887(micr);
            Code.BackColor = SystemColors.Control;
            Code.ReadOnly = true;
            Save_transition_labels(Code.Text);
            Intermediate_translate(Code.Text);
            Mark_Saving_command();
            addr = 0;
            stop = false;
            Pointer_To_ROM.Location = new Point(0,
Pointer_To_ROM.Location.Y - Point_To_ROM_Location * 20);
            Pointer_To_Code.Location = new Point(0,
Pointer_To_Code.Location.Y - Point_To_Code_Location * 20);
            Point_To_Code_Location = 0;
            Point_To_ROM_Location = 0;
            //Color_the_text();
        }
        private void Color_the_text()
        {
            int s = Code.SelectionStart;

            string[] Code_strs = Code.Text.Split("\n");
            int indent = 0;
            for (int i = 0; i < Code_strs.Length; i++)
            {
                Listing_ROM.Text = "";
                MatchCollection Texts = new
Regex(@"\S+").Matches(Code_strs[i]);
                MatchCollection Spaces = new
Regex(@"\s+").Matches(Code_strs[i]);
                int Spaces_add_ind = 0;
                int Texts_add_ind = 0;
```

```
                if (Regex.IsMatch(Code_strs[i], "^\\s+"))
                    indent += Spaces[Spaces_add_ind++].Length;
                while (Spaces.Count() + Texts.Count() >
Spaces_add_ind + Texts_add_ind)
                {
                    if (Texts_add_ind != Texts.Count())
                    {
                        Code.SelectionStart = indent;
                        Code.SelectionLength =
Texts[Texts_add_ind].Value.Length;
                        switch
(micr.Key_designations(Texts[Texts_add_ind].Value.ToUpper()))
                        {
                            case "comand":
                                Code.SelectionColor =
Color.DarkCyan;
                                break;
                            case "name_registers":
                                Code.SelectionColor =
Color.Purple;
                                break;
                        }
                        indent +=
Texts[Texts_add_ind++].Length;
                    }
                    if (Spaces_add_ind != Spaces.Count())
                        indent +=
Spaces[Spaces_add_ind++].Length;
                }
                indent++;
            }
            Code.SelectionStart = s;
            Code.SelectionLength = 0;
            Code.SelectionColor = Color.Black;
```

```
        }
        private void Compilation_Click(object sender,
EventArgs e)
        {
            if (Compilation.Text == "Stop debug")
            {
                Compilation.Text = "Compilation";
                Code.ReadOnly = false;
                Code.BackColor = SystemColors.Window;
                Listing_ROM.Text = "";
                stop = true;
                Pointer_To_Code.Visible = false;
                Pointer_To_ROM.Visible = false;
                if (!command_processing)
                {
                    Start.Enabled = false;
                    Step.Enabled = false;
                    Execution.Time_of_running = 0;
                    Time.Text = "Time: " +
Execution.Time_of_running;
                }
            }
            else
            {
                microcontroller_preparation();
                if (Errors == 0)
                {
                    Compilation.Text = "Stop debug";
                    Debug_console.Text += "Ошибок не
обнаружено\n";
                    Start.Enabled = true;
                    Step.Enabled = true;
                    Start.Text = "Start";
                    Pointer_To_Code.Visible = true;
```

```
                    Pointer_To_ROM.Visible = true;
                }
                else
                {
                    Debug_console.Text += "Ошибок обнаружено ("
+ Errors + ")\n";
                    Compilation.Text = "Compilation";
                    Code.ReadOnly = false;
                    Code.BackColor = SystemColors.Window;
                    Listing_ROM.Text = "";
                }
            }

        }

        //выполнение
        private void Start_Click(object sender, EventArgs e)
        {
            if (Start.Text == "Start")
            {
                command_processing = true;
                stop = false;
                Step.Enabled = false;
                Start.Text = "Stop";
                Execution.myThread = new Thread(Run_commands);
                Execution.myThread.Start();
            }
            else
            {
                stop = true;
            }
        }
        public void Run_commands()
        {
```

```
            try
            {
                while (!stop)
                {
                    int addr = Execution.Run_command(micr);
                    Time.Invoke(new Action<int>((b) =>
Point_update(b)), addr);
                    if
(line_to_command.ContainsKey(Point_To_ROM_Location))
                        if
(Breskpoints.IndexOf(line_to_command[Point_To_ROM_Location])
>= 0)
                            stop = true;
                    if (!stop)
                        System.Threading.Thread.Sleep(delay);
                    Time.Invoke(new Action<decimal>((b) =>
Time.Text = "Time: " + ("" + b).Substring(0, 10)),
Execution.Time_of_running);
                }
                if (Compilation.Text == "Stop debug")
                {
                    Start.Invoke(new Action<string>((b) =>
Start.Text = b), "Start");
                    Step.Invoke(new Action<bool>((b) =>
Step.Enabled = b), true);
                }
                else
                {
                    Start.Invoke(new Action<bool>((b) =>
Start.Enabled = b), false);
                    Step.Invoke(new Action<bool>((b) =>
Step.Enabled = b), false);
                    Execution.Time_of_running = 0;
                    Time.Invoke(new Action<decimal>((b) =>
```

```
Time.Text = "Time: " + b), Execution.Time_of_running);
                }
                command_processing = false;
            }
            catch (Exception e) { }
        }
        private void Step_Click(object sender, EventArgs e)
        {
            int addr = Execution.Run_command(micr);
            Point_update(addr);
            Time.Text = "Time: " + ("" +
Execution.Time_of_running).Substring(0, 10);
            Start.Enabled = true;
        }
        public void Point_update(int addr)
        {
            if (addr >= 0)
            {
                Point_To_ROM_Location = addr;
                if
(line_to_command.ContainsKey(Point_To_ROM_Location))
                    Point_To_Code_Location =
line_to_command[Point_To_ROM_Location] - 1;
            }
            int Pos_Code = GetScrollPos(Code.Handle,
(int)System.Windows.Forms.Orientation.Vertical);
            int Pos_Listing_ROM =
GetScrollPos(Listing_ROM.Handle,
(int)System.Windows.Forms.Orientation.Vertical);
            if
(line_to_command.ContainsKey(Point_To_ROM_Location))
            {
                Point_To_Code_Location =
line_to_command[Point_To_ROM_Location] - 1;
                Pointer_To_Code.Location = new Point(0,
```

```
Point_To_Code_Location * 20 - Pos_Code);
            }
            Pointer_To_ROM.Location = new Point(0,
Point_To_ROM_Location * 20 - Pos_Listing_ROM);
        }

        //периферия
        private void button_Click(object sender, EventArgs e)
        {
            Button This_button = (Button)sender;
            ComboBox[] ComboBox_buttons = new ComboBox[] {
comboBox_button1, comboBox_button2, comboBox_button3,
comboBox_button4 };
            Button[] Buttons = new Button[] { button1,
button2, button3, button4 };
            int index =
ComboBox_buttons[Array.IndexOf(Buttons,
This_button)].SelectedIndex;
            if (index <= 0)
                return;
            if (This_button.Text == "OFF")
            {
                This_button.Text = "ON";
                Execution.Input_values_from_buttons(micr,
index - 1, 1);
            }
            else
            {
                This_button.Text = "OFF";
                Execution.Input_values_from_buttons(micr,
index - 1, 0);
            }
            if (Start.Text != "Start")
                return;
```

```
        }
        private void comboBox_led1_SelectedIndexChanged(object
sender, EventArgs e)
        {
            leds = comboBox_led1.SelectedIndex;
            PictureBox[] pictureBox = { led8, led7, led6,
led5, led4, led3, led2, led1 };
            if (leds == 0)
                foreach (PictureBox led in pictureBox)
                    led.BackColor = Color.Gray;
            if (Start.Text != "Start")
                return;
        }
        private void Update_leds(PIC16f887 mic)
        {
            if (leds == 0) return;
            PictureBox[] pictureBox = { led8, led7, led6,
led5, led4, led3, led2, led1 };
            for (int i = 0; i < 8; i++)
            {
                switch
(Execution.Output_pin_digital_request(mic, leds - 1, i))
                {
                    case 0:
                        pictureBox[i].BackColor =
Color.DarkRed;
                        break;
                    case 1:
                        pictureBox[i].BackColor =
Color.DarkGreen;
                        break;
                    default:
                        pictureBox[i].BackColor = Color.Gray;
                        break;
                }
```

```
            }
        }

        //таблица
        private void Add_FSR_Click(object sender, EventArgs e)
        {
            int adress =
Execution.Find_out_the_register_addres(micr,
Register_display.Text);
            mutexObj.WaitOne();
            Table[Table.Count - 1] =
Execution.Find_out_the_register_name(micr, adress);
            Table.Add("");
            mutexObj.ReleaseMutex();
        }
        private TextBox[] Create_a_string()
        {
            TextBox[] line = new TextBox[5];
            if (liner_Size[0, 0] == 0)
            {
                int ind = 0;
                while (ind < View_RAM.Controls.Count)
                {
                    if (View_RAM.Controls[ind].Text ==
"Aders")
                        break;
                    ind++;
                }
                for (int i = 0; i < 5; i++)
                {
                    liner_Size[i, 0] = View_RAM.Controls[ind +
i].Size.Width;
                    liner_Size[i, 1] = View_RAM.Controls[ind +
i].Size.Height;
```

```
                    table_width += View_RAM.Controls[ind +
i].Size.Width;
                }
                liner_Size[0, 0] -= 2;
                liner_Size[4, 0] -= 2;
            }
            for (int i = 0; i < 5; i++)
            {
                line[i] = new TextBox();
                line[i].RightToLeft = RightToLeft.No;
                line[i].Size = new Size(liner_Size[i, 0],
liner_Size[i, 1]);
                line[i].Margin = new Padding(0);
                line[i].TabIndex = 0;
                line[i].ReadOnly = true;
                line[i].TextAlign = HorizontalAlignment.Right;
                line[i].TabStop = false;
                line[i].MaxLength = 8;
                //line[i].de
                line[i].KeyDown +=
Deleting_a_row_from_a_table;
                if (i < 2)
                {
                    line[i].DoubleClick +=
Call_field_to_update_table;
                    line[i].KeyPress +=
Call_field_to_update_table;
                }
            }
            return line;
        }
        private void Deleting_a_row_from_a_table(object
sender, EventArgs e)
        {
            TextBox? textBox = sender as TextBox;
```

```
            if (e is KeyEventArgs)
            {
                KeyEventArgs t = (KeyEventArgs)e;
                if (t.KeyData == Keys.Delete)
                {
                    int dell_ind =
Register_status.Controls.IndexOf(textBox) / 5;
                    if (dell_ind == Table.Count - 1)
                        return;
                    Table.RemoveAt(dell_ind);
                    /*if (!command_processing)
                        Update_Table();*/
                }
                return;
            }
        }
        private void Call_field_to_update_table(object sender,
EventArgs e)
        {
            TextBox? textBox = sender as TextBox;
            textBox_Input = new TextBox();
            if (e is MouseEventArgs)
            {
                textBox_Input.Text = textBox.Text;
            }
            if (e is KeyPressEventArgs)
            {
                KeyPressEventArgs t = (KeyPressEventArgs)e;
                textBox_Input.Text += t.KeyChar;
            }

            textBox_Input.RightToLeft = RightToLeft.No;
            textBox_Input.Size = textBox.Size;
            textBox_Input.Margin = new Padding(0);
            textBox_Input.TabIndex = 0;
```

```
            textBox_Input.TextAlign =
HorizontalAlignment.Right;
            textBox_Input.TabStop = false;
            textBox_Input.MaxLength = 14;
            textBox_Input.Location = new
Point(textBox.Location.X + View_RAM.Location.X +
Register_status.Location.X + 2, textBox.Location.Y +
View_RAM.Location.Y + Register_status.Location.Y + 2); //нужный
указатель
            Controls.Add(textBox_Input);
            textBox_Input.BringToFront();
            textBox_Input.Focus();
            textBox_Input.Select(textBox_Input.Text.Length,
0);
            textBox_Input.Leave += Table_diction;
            index = Register_status.Controls.IndexOf(textBox);
        }
        private void Table_diction(object sender, EventArgs e)
        {
            TextBox box = (TextBox)sender;
            string Text = box.Text.ToUpper();
            Controls.Remove(textBox_Input);
            int adress;
            switch (index % 5)
            {
                case 0:
                    if (!Regex.IsMatch(Text, "^\\s*([0-
9]|A|B|C|D|E|F)+\\s*$"))
                        return;
                    else
                        if (!(Convert.ToInt32(Text, 16) > 0 &&
Convert.ToInt32(Text, 16) < 512))
                        return;
                    adress = Convert.ToInt32(Text, 16);
```

```
                    break;
                case 1:
                    adress =
Execution.Find_out_the_register_addres(micr, Text);
                    if (adress == -1)
                        return;
                    break;
                default:
                    return;
            }

            string name =
Execution.Find_out_the_register_name(micr, adress);
            mutexObj.WaitOne();
            Table[index / 5] = name;
            if (Table.Count - 1 == index / 5)
                Table.Add("");
            mutexObj.ReleaseMutex();
        }
        private void Update_Table(PIC16f887 mic)
        {
            mutexObj.WaitOne();
            List<string> lines = new List<string>(Table);
            mutexObj.ReleaseMutex();
            for (int i = Register_status.Controls.Count / 5; i
< lines.Count; i++)
            {
                TextBox[] tt = Create_a_string();
                foreach (TextBox t in tt)
                    Register_status.Invoke(new
Action<TextBox>((str) => Register_status.Controls.Add(str)),
t);
            }
            int range = Register_status.Controls.Count;
            for (int i = lines.Count * 5; i < range; i++)
```

```
                Register_status.Invoke(new Action<bool>((b) =>
Register_status.Controls.RemoveAt(Register_status.Controls.Cou
nt - 1)), true);

            if (Register_status.Controls.Count < 35)
                Register_status.Invoke(new Action<int>((i) =>
Register_status.Size = new Size(i, 180)), table_width);
            else
                Register_status.Invoke(new Action<int>((i) =>
Register_status.Size = new Size(i + 22, 180)), table_width);

            for (int i = 0; i < lines.Count; i++)
            {
                if (lines[i] == "")
                {
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5].Text =
str), i, "");
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 1].Text =
str), i, "");
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 2].Text =
str), i, "");
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 3].Text =
str), i, "");
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 4].Text =
str), i, "");
                    continue;
                }
                int adress =
Execution.Find_out_the_register_addres(mic, lines[i]);
                int value =
```

```
Execution.Find_out_the_register_status(mic, adress);
                int n_bits = 8;
                if ((value & 65536) != 0)
                    n_bits = 16;
                value = value & 65535;
                if (adress > 511)
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5].Text =
str), i, "");
                else
                    Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5].Text =
str), i, Convert.ToString(adress, 16).PadLeft(3,
'0').ToUpper());
                Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 1].Text =
str), i, lines[i]);
                Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 2].Text =
str), i, "0x" + Convert.ToString(value,
16).ToUpper().PadLeft(n_bits / 4, '0'));
                Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 3].Text =
str), i, Convert.ToString(value, 10));
                Register_status.Invoke(new Action<int,
string>((i, str) => Register_status.Controls[i * 5 + 4].Text =
str), i, Convert.ToString(value, 2).PadLeft(n_bits, '0'));
            }
        }
        private void Update_data_Output()
        {
            while (update)
            {
                try
                {
```

```
                    System.Threading.Thread.Sleep(1);
                    PIC16f887 mic = micr;
                    string t =
Execution.Update_data_on_ports(mic);
                    if(t != "")
                        Register_status.Invoke(new
Action<string>((b) => Debug_console.Text += b), t);
                    Update_leds(micr);
                    Update_Table(micr);
                    Update_Stack(micr);
                }
                catch (Exception e) { }
            }
        }
        private void Delay_formation(object sender, EventArgs
e)
        {
            if (Regex.IsMatch(Delay.Text, "^[0-9]+$") &&
!Regex.IsMatch(Delay.Text, "^0+$"))
                delay = Convert.ToInt32(Delay.Text);
            else
                Delay.Text = Convert.ToString(delay);
        }

        //Стек
        private void Create_a_Stack()
        {
            for (int n = 0; n < 8; n++)
            {
                Label[] line = new Label[2];
                for (int i = 0; i < 2; i++)
                {
                    line[i] = new Label();
                    line[i].RightToLeft = RightToLeft.No;
```

```
                    line[i].Margin = new Padding(0);
                    line[i].TabIndex = 0;
                    line[i].TabStop = false;
                    line[i].TextAlign =
ContentAlignment.MiddleRight;
                    line[i].BackColor =
SystemColors.ActiveBorder;
                    line[i].BorderStyle = BorderStyle.Fixed3D;
                }
                line[1].Size = new
Size(Stack_Output.Controls[0].Size.Width - 2,
Stack_Output.Controls[0].Size.Height - 2);
                line[0].Size = new
Size(Stack_Output.Controls[1].Size.Width - 2,
Stack_Output.Controls[1].Size.Height - 2);
                line[0].Text = Convert.ToString(n);
                line[1].Text = Convert.ToString(0);
                Stack_view.Controls.AddRange(line);
            }
        }
        private void Update_Stack(PIC16f887 mic)
        {
            ushort[] data = mic.Stack_output();
            for (int i = 0; i < data.Length; i++)
                Stack_view.Invoke(new Action<string>((data) =>
Stack_view.Controls[2 * i + 1].Text = data),
Convert.ToString(data[i]));
        }

        //для  отладки
        private Label Creat_Breakpoint()
        {
            Label point = new Label();
            point.AutoSize = false;
```

```
            point.Size = new Size(23, 20);
            point.Location = new Point(-5, 20 *
BreakpointsView.Controls.Count);
            point.TabStop = false;
            point.ForeColor = Color.Red;
            point.TextAlign = ContentAlignment.MiddleLeft;
            point.MouseDown += Breakpoint_MouseDown;
            point.MouseEnter += Breakpoint_MouseEnter;
            point.MouseLeave += Breakpoint_MouseLeave;
            return point;
        }
        private void Breakpoint_MouseDown(object sender,
EventArgs e)
        {
            Label point = sender as Label;
            if (point.ForeColor == SystemColors.GrayText &&
BreakpointsView.Controls.IndexOf(point) >= 0)
            {
                point.ForeColor = Color.Red;

Breskpoints.Add(Convert.ToInt32(Numbering.Controls[Breakpoints
View.Controls.IndexOf(point)].Text));
            }
            else
            {
                point.ForeColor = SystemColors.GrayText;

Breskpoints.Remove(Convert.ToInt32(Numbering.Controls[Breakpoi
ntsView.Controls.IndexOf(point)].Text));
            }
        }
        private void Breakpoint_MouseEnter(object sender,
EventArgs e)
        {
            Label t = sender as Label;
```

```
            if (t.Text == "")
            {
                t.ForeColor = SystemColors.GrayText;
                t.Text = "●";
            }
        }
        private void Breakpoint_MouseLeave(object sender,
EventArgs e)
        {
            Label t = sender as Label;
            if (t.ForeColor == SystemColors.GrayText)
            {
                t.Text = "";
            }
        }
        private Label Creat_Numbering()
        {
            Label point = new Label();
            point.AutoSize = false;
            point.Size = new Size(Numbering.Size.Width, 20);
            point.Location = new Point(0, 20 *
Numbering.Controls.Count);
            point.TabStop = false;
            point.Text = "" + (Numbering.Controls.Count + 1);
            point.TextAlign = ContentAlignment.MiddleRight;
            return point;
        }
        private void Numbering_up(int Pos)
        {
            int Code_N_str = Code.Text.Count(c => c == '\n') +
1;
            Pointer_To_Code.Location = new Point(0,
Point_To_Code_Location * 20 - Pos);
            for (int i = 0; i < Numbering.Controls.Count; i++)
```

```
            {
                int p_Numb =
Convert.ToInt32(Numbering.Controls[i].Text) - 1;
                if (i + 1 > Code_N_str)
                {
                    Numbering.Controls[i].Visible = false;
                    BreakpointsView.Controls[i].Visible =
false;
                    Numbering.Controls[i].Enabled = false;
                    BreakpointsView.Controls[i].Enabled =
false;

if(Breskpoints.IndexOf(Convert.ToInt32(Numbering.Controls[i].T
ext)) >= 0)

Breskpoints.Remove(Convert.ToInt32(Numbering.Controls[i].Text)
);
                    BreakpointsView.Controls[i].ForeColor =
SystemColors.GrayText;
                    BreakpointsView.Controls[i].Text = "";
                }
                else
                {
                    Numbering.Controls[i].Visible = true;
                    BreakpointsView.Controls[i].Visible =
true;
                    Numbering.Controls[i].Enabled = true;
                    BreakpointsView.Controls[i].Enabled =
true;
                }
                Numbering.Controls[i].Location = new Point(0,
20 * p_Numb - Pos);
                BreakpointsView.Controls[i].Location = new
Point(-5, 20 * p_Numb - Pos);
```

```
                while (Numbering.Controls[i].Location.Y < -
320)
                {
                    Numbering.Controls[i].Text = "" +
(Convert.ToInt32(Numbering.Controls[i].Text) + 64);
                    Numbering.Controls[i].Location = new
Point(0, Numbering.Controls[i].Location.Y + 1280);
                    BreakpointsView.Controls[i].Location = new
Point(-5, BreakpointsView.Controls[i].Location.Y + 1280);
                    if
(Breskpoints.IndexOf(Convert.ToInt32(Numbering.Controls[i].Tex
t)) >= 0)
                    {
                        BreakpointsView.Controls[i].ForeColor
= Color.Red;
                        BreakpointsView.Controls[i].Text =
"●";
                    }
                    else
                    {
                        BreakpointsView.Controls[i].ForeColor
= SystemColors.GrayText;
                        BreakpointsView.Controls[i].Text = "";
                    }
                }
                while (Numbering.Controls[i].Location.Y > 960)
                {
                    Numbering.Controls[i].Text = "" +
(Convert.ToInt32(Numbering.Controls[i].Text) - 64);
                    Numbering.Controls[i].Location = new
Point(0, Numbering.Controls[i].Location.Y - 1280);
                    BreakpointsView.Controls[i].Location = new
Point(-5, BreakpointsView.Controls[i].Location.Y - 1280);
```

```
                    if
(Breskpoints.IndexOf(Convert.ToInt32(Numbering.Controls[i].Tex
t)) >= 0)
                    {
                        BreakpointsView.Controls[i].ForeColor
= Color.Red;
                        BreakpointsView.Controls[i].Text =
"●";
                    }
                    else
                    {
                        BreakpointsView.Controls[i].ForeColor
= SystemColors.GrayText;
                        BreakpointsView.Controls[i].Text = "";
                    }
                }
            }
        }
        private void Listing_ROM_VScroll(object sender,
EventArgs e)
        {
            RichTextBox Box = sender as RichTextBox;
            int Pos_Listing_ROM = GetScrollPos(Box.Handle,
(int)System.Windows.Forms.Orientation.Vertical);
            Pointer_To_ROM.Location = new Point(0,
Point_To_ROM_Location * 20 - Pos_Listing_ROM);
        }
        private void Form_Closing(object sender,
FormClosingEventArgs e)
        {
            stop = true;
            update = false;
        }
        private void Remove_breakpoints_Click(object sender,
```

```
EventArgs e)
        {
            Breskpoints.Clear();
            foreach (Label br in BreakpointsView.Controls)
            {
                br.ForeColor = SystemColors.GrayText;
                br.Text = "";
            }
        }
    }
}
```

Файл Execution.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using static System.Net.Mime.MediaTypeNames;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.TaskbarCl
ock;

namespace dip
{
    struct data_result_alu
    {
        public bool NOP_flg;
        public byte result;
```

```
        public byte new_status;

        public data_result_alu(byte result, byte new_status,
bool NOP_flg = false)
        {
            this.result = result;
            this.new_status = new_status;
            this.NOP_flg = NOP_flg;
        }
    };
    struct Event
    {
        public int pin;
        public int mode;
        public Event(int pin, int mode)
        {
            this.pin = pin;
            this.mode = mode;
        }
    };

    internal static class Execution
    {
        static public Thread? myThread;
        static public Thread? table_update;
        static int cycle;
        static Mutex mutexObj = new Mutex();
        static bool flg_RBIF = false;
        static List<Event> Events = new List<Event>();
        static public decimal Time_of_running = 0;
        static public string Update_data_on_ports(PIC16f887
mic)//обновление портов
        {
            string output = "";
```

```
            mutexObj.WaitOne();
            Event[] Now_Events = new Event[Events.Count];
            Events.CopyTo(Now_Events);
            Events.Clear();
            mutexObj.ReleaseMutex();

            foreach (Event e in Now_Events)
            {
                int port = e.pin / 8 + 5;
                int bit = e.pin % 8;
                if (e.mode == 0 | e.mode == 1) //обработка
цифового ввода
                {
                    bool TRIS = true, ANSEL = true;
                    int Mask_porta = mic.Read_RAM(392);
                    int Mask_portb = mic.Read_RAM(393);
                    int Mask_porte = Mask_porta;

                    Mask_porta = (Mask_porta & 15) |
((Mask_portb & 16) >> 1);
                    Mask_portb = ((Mask_portb & 16) >> 4) |
((Mask_portb & 4) >> 1) | ((Mask_portb & 1) << 2) |
((Mask_portb & 2) << 2) | ((Mask_portb & 8) << 1) |
((Mask_portb & 32));
                    Mask_porte = (Mask_porte >> 5);

                    if ((mic.Read_RAM(133 + e.pin / 8) & (1 <<
bit)) == 0)
                    { //TRIS
                        output += mic.pins[e.pin] + " pin
configured a an output, requires setup (TRIS" +
mic.pins[e.pin][1] + ").\n";
                        TRIS = false;
                    }
```

```
                    switch (port)
                    {
                        case 5:
                            if (((Mask_porta >> bit) & 1) ==
1)
                            { //ANSEL
                                output += mic.pins[e.pin] + "
pin is assigned as analog input, requires setup (ANSEL).\n";
                                ANSEL = false;
                            }
                            break;
                        case 6:
                            if (((Mask_portb >> bit) & 1) ==
1)
                            {
                                output += mic.pins[e.pin] + "
pin is assigned as analog input, requires setup (ANSELH).\n";
                                ANSEL = false;
                            }
                            break;
                        case 9:
                            if (((Mask_porte >> bit) & 1) ==
1)
                            {
                                output += mic.pins[e.pin] + "
pin is assigned as analog input, requires setup (ANSELH).\n";
                                ANSEL = false;
                            }
                            break;
                    }
                    if (TRIS)
                        if (ANSEL)
                            mic.ports[e.pin] = e.mode;
                        else
```

```
                            mic.ports[e.pin] = 0;
                    }
                    if (e.mode < 0) //обработка аналогового ввода
                    {
                        mic.ports[e.pin] = 0;
                    }
                }
                return output;
            }
            static private void Loading_data_from_ports(PIC16f887
mic)
            {
                int port_n = 0;
                for (int i = 0; i < mic.ports.Length; i++)
                {
                    if(mic.ports[i] == 1 || mic.ports[i] == 0) //
только цифровой сигнал
                        port_n += mic.ports[i] << (i % 8);

                    if (i % 8 == 7 || i == 35)
                    {
                        int port_befor = mic.Read_RAM(5 + i / 8) &
~mic.Read_RAM(133 + i / 8);
                        int port_now = port_n & mic.Read_RAM(133 +
i / 8);

                        if (i / 8 == 1) {
                            if ((((mic.Read_RAM(6) &
mic.Read_RAM(134)) ^ port_now) & mic.Read_RAM(150)) != 0)
                                flg_RBIF = true;
                            if ((((mic.Read_RAM(129) & 64) == 64)
&& ((mic.Read_RAM(6) & 1) == 0) && (port_now & 1) == 1) ||
(((mic.Read_RAM(129) & 64) == 0) && ((mic.Read_RAM(6) & 1) ==
1) && ((port_now & 1) == 0)))
```

```
                            mic.Write_RAM(11,
(byte)(mic.Read_RAM(11) | 2));
                    }
                    mic.Write_RAM(5 + i / 8, (byte)(port_befor
| port_now));
                    port_n = 0;
                }
            }
        }
        static private void Loading_data_on_ports(PIC16f887
mic)
        {
            int Mask_porta = mic.Read_RAM(392);
            int Mask_portb = mic.Read_RAM(393);
            int Mask_porte = Mask_porta;

            Mask_porta = (Mask_porta & 15) | ((Mask_portb &
16) >> 1);
            Mask_portb = ((Mask_portb & 16) >> 4) |
((Mask_portb & 4) >> 1) | ((Mask_portb & 1) << 2) |
((Mask_portb & 2) << 2) | ((Mask_portb & 8) << 1) |
((Mask_portb & 32));
            Mask_porte = (Mask_porte >> 5);

            int port = 0;
            for (int i = 0; i < mic.ports.Length; i++)
            {
                int n_port = i / 8;
                int bit = (mic.Read_RAM(5 + n_port) >> (i %
8)) & 1;
                int bit2 = (mic.Read_RAM(5 + n_port) >> (i %
8));
                int bit3 = mic.Read_RAM(5 + n_port);
                if ((mic.Read_RAM(133 + n_port) >> (i & 8)) ==
```

```
1)//TRIS = 1 - не трогаем
                    continue;

                if (((Mask_porta >> i) & 1) == 1 && n_port ==
0)//ANSEL
                {
                    if (mic.ports[i] >= 0)
                        mic.ports[i] = -1;
                    continue;
                }

                if (((Mask_portb >> i) & 1) == 1 && n_port ==
1)//ANSEL
                {
                    if (mic.ports[i] >= 0)
                        mic.ports[i] = -1;
                    continue;
                }

                if (((Mask_porte >> i) & 1) == 1 && n_port ==
5)//ANSEL
                {
                    if(mic.ports[i] >= 0)
                        mic.ports[i] = -1;
                    continue;
                }

                mic.ports[i] = bit;
            }
        }
        static public int Run_command(PIC16f887 mic)
        {
            flg_RBIF = false;
            Loading_data_from_ports(mic);
```

```
            cycle = 1;
            data_result_alu result;
            result.NOP_flg = false;
            if ((mic.Read_RAM(3) & 8) == 0) //проверка на SLEEP
            {
                bool inter = Check_inclusion(mic);
                if (inter)
                {
                    mic.CLRWDT();
                    mic.Write_RAM(3, (byte)(mic.Read_RAM(3) |
8));
                }
                return ((mic.Read_RAM(10) & 31) << 8 |
mic.Read_RAM(2));
            }
            ushort instruction_reg = Reading_command(mic);
            byte contents_register;
            int adde_register;
            switch (instruction_reg)
            {
                case 8: //RETURN
                case 9: //RETFIE
                    if (instruction_reg == 9)
                        mic.Write_RAM(11,
(byte)(mic.Read_RAM(11) | 128));
                    ushort PC = mic.Pop();
                    mic.Write_RAM(2, (byte)PC);
                    mic.Write_RAM(10, (byte)(PC >> 8));
                    cycle = 2;
                    break;

                case 99: //SLEEP //потом реализовать
                    if(Check_inclusion(mic))
                        break;
```

```
                    mic.CLRWDT();
                    mic.Write_RAM(3, (byte)((mic.Read_RAM(3) &
~8) | 16));
                    break;

                case 100: //CLRWDT
                    mic.CLRWDT();
                    if (instruction_reg == 100)
                    mic.Write_RAM(3, (byte)(mic.Read_RAM(3) |
24));
                    break;
                default:
                    switch (instruction_reg >> 12)
                    {
                        case 0: //BYTE-ORIENTED FILE REGISTER
OPERATIONS
                            adde_register = Adde_MUX(mic,
(byte)(instruction_reg & 127));
                            contents_register =
mic.Read_RAM(adde_register);
                            result = ALU_BYTE(mic,
(byte)((instruction_reg >> 8) & 15), contents_register);
                            mic.Write_RAM(3,
result.new_status);

                            if (((instruction_reg >> 7) & 1)
== 1) {
                                mic.Write_RAM(adde_register,
result.result, true);
                            }
                            else
                                mic.Wreg = result.result;

                            break;
```

```
                        case 1: //BIT-ORIENTED FILE REGISTER
OPERATIONS
                            adde_register = Adde_MUX(mic,
(byte)(instruction_reg & 127));
                            contents_register =
mic.Read_RAM(adde_register);
                            result =
ALU_BITE((byte)((instruction_reg >> 10) & 3),
(byte)((instruction_reg >> 7) & 7), contents_register);
                            mic.Write_RAM(adde_register,
result.result, true);
                            break;

                        case 2: //LITERAL AND CONTROL
OPERATIONS
                            ALU_transitions(mic,
instruction_reg); //2 цикла
                            break;

                        case 3:
                            result = ALU_other(mic,
instruction_reg, (byte)(instruction_reg & 255));
                            mic.Wreg = result.result;
                            break;
                    }
                    break;
            }
            if (result.NOP_flg)
            {
                Time_of_running += mic.Time_of_1_cycle(1);
                Reading_command(mic);
            }

            int newcycle = 0;
```

```
            while (!Go_to_inclusion(mic, Check_inclusion(mic))
&& newcycle < cycle){
                Time_of_running += mic.Time_of_1_cycle(1);
                newcycle++;
            }
            Loading_data_on_ports(mic);



            return ((mic.Read_RAM(10) & 31) << 8 |
mic.Read_RAM(2));
        }
        static private ushort Reading_command(PIC16f887 mic)
        {
            ushort PCL = mic.Read_RAM(2);
            ushort PCLATH = mic.Read_RAM(10);
            ushort PC = (ushort)((PCLATH & 31) << 8 | PCL);
            ushort cmd = mic.Read_ROM(PC);
            PC++;
            if (PC == mic.Take_numb_cmd())
                PC = 0;
            mic.Write_RAM(2, (byte)PC);
            mic.Write_RAM(10, (byte)(PC >> 8));
            return cmd;
        }
        static private void ALU_transitions(PIC16f887 mic,
ushort cmd)
        {
            ushort PC, PCL;
            ushort PCLATH = mic.Read_RAM(10);
            if ((cmd >> 11 & 1) == 0) //CALL
            {
                PCL = mic.Read_RAM(2);
                PC = (ushort)(((PCLATH & 31) << 8) | PCL);
                mic.Push(PC);
```

```
            }
            PC = (ushort)(((PCLATH & 24) << 8) | (cmd &
2047)); //GOTO
            mic.Write_RAM(2, (byte)PC);
            mic.Write_RAM(10, (byte)(PC >> 8));
            cycle = 2;
        }
        static private data_result_alu ALU_other(PIC16f887
mic, ushort cmd, byte operand)
        {
            switch (cmd >> 8)
            {
                case 62: //ADDLW
                    return new data_result_alu((byte)(operand
+ mic.Wreg), (byte)(mic.Read_RAM(3) & 248 |
Create_Status(operand + mic.Wreg, operand & 15 + mic.Wreg &
15) & 7));
                case 57: //ANDLW
                    return new data_result_alu((byte)(operand
& mic.Wreg), (byte)(mic.Read_RAM(3) & 251 |
Create_Status(operand & mic.Wreg) & 4));
                case 56: //IORLW
                    return new data_result_alu((byte)(operand
| mic.Wreg), (byte)(mic.Read_RAM(3) & 251 |
Create_Status(operand | mic.Wreg) & 4));
                case 48: //MOVLW
                    return new data_result_alu(operand,
mic.Read_RAM(3));
                case 52: //RETLW
                    mic.Wreg = operand;
                    ushort PC = mic.Pop();
                    mic.Write_RAM(2, (byte)PC);
                    mic.Write_RAM(10, (byte)(PC >> 8));
                    cycle = 2;
                    return new data_result_alu(operand,
```

```
mic.Read_RAM(3));
                case 60: //SUBLW
                    return new data_result_alu((byte)(operand
- mic.Wreg), (byte)(mic.Read_RAM(3) & 248 |
Create_Status(operand - mic.Wreg, operand & 15 - mic.Wreg &
15) & 7));
                case 58: //XORLW
                    return new data_result_alu((byte)(operand
^ mic.Wreg), (byte)(mic.Read_RAM(3) & 251 |
Create_Status(operand ^ mic.Wreg) & 4));
            }
            return new data_result_alu(0, 0);
        }
        static private data_result_alu ALU_BYTE(PIC16f887 mic,
byte cmd, byte contents_register)
        {
            switch (cmd)
            {
                case 0: //MOVWF/NOP
                    return new data_result_alu(mic.Wreg,
mic.Read_RAM(3));

                case 1: //CLRF/CLRW
                    return new data_result_alu(0,
(byte)(mic.Read_RAM(3) & 251));

                case 2: //SUBWF
                    return new
data_result_alu((byte)(contents_register - mic.Wreg),
(byte)(mic.Read_RAM(3) & 248 | Create_Status(contents_register
- mic.Wreg, contents_register & 15 - mic.Wreg & 15) & 7));

                case 3: //DECF
                    return new
data_result_alu((byte)(contents_register - 1),
```

```
(byte)(mic.Read_RAM(3) & 251 | Create_Status(contents_register
- 1) & 4));

                case 4: //IORWF
                    return new
data_result_alu((byte)(contents_register | mic.Wreg),
(byte)(mic.Read_RAM(3) & 251 | Create_Status(contents_register
| mic.Wreg) & 4));

                case 5: //ANDWF
                    return new
data_result_alu((byte)(contents_register & mic.Wreg),
(byte)(mic.Read_RAM(3) & 251 | Create_Status(contents_register
& mic.Wreg) & 4));

                case 6: //XORWF
                    return new
data_result_alu((byte)(contents_register ^ mic.Wreg),
(byte)(mic.Read_RAM(3) & 251 | Create_Status(contents_register
^ mic.Wreg) & 4));

                case 7: //ADDWF
                    return new
data_result_alu((byte)(contents_register + mic.Wreg),
(byte)(mic.Read_RAM(3) & 248 | Create_Status(contents_register
+ mic.Wreg, contents_register & 15 + mic.Wreg & 15) & 7));

                case 8: //MOVF
                    return new
data_result_alu((byte)(contents_register),
(byte)(mic.Read_RAM(3) & 251 |
Create_Status(contents_register) & 4));

                case 9: //COMF
                    return new
```

```
data_result_alu((byte)(~contents_register),
(byte)(mic.Read_RAM(3) & 251 |
Create_Status(~contents_register) & 4));

                case 10: //INCF
                    return new
data_result_alu((byte)(contents_register + 1),
(byte)(mic.Read_RAM(3) & 251 | Create_Status(contents_register
+ 1) & 4));

                case 11: //DECFSZ
                    if (contents_register - 1 == 0)
                        return new
data_result_alu((byte)(contents_register - 1),
mic.Read_RAM(3), true);
                    else
                        return new
data_result_alu((byte)(contents_register - 1),
mic.Read_RAM(3));

                case 12: //RRF
                    return new
data_result_alu((byte)((contents_register << 1) |
(mic.Read_RAM(3) & 1)), (byte)(mic.Read_RAM(3) & 254 |
(contents_register >> 7)));

                case 13: //RLF
                    return new
data_result_alu((byte)((contents_register >> 1) |
((mic.Read_RAM(3) & 1) << 7)), (byte)(mic.Read_RAM(3) & 254 |
contents_register & 1));

                case 14: //SWAPF
                    return new
data_result_alu((byte)(((contents_register << 4) |
```

```
(contents_register >> 4))), mic.Read_RAM(3));

                case 15: //INCFSZ
                    if (contents_register + 1 == 0)
                        return new
data_result_alu((byte)(contents_register + 1),
mic.Read_RAM(3), true); //добавить NOP потом
                    else
                        return new
data_result_alu((byte)(contents_register + 1),
mic.Read_RAM(3));
            }
            return new data_result_alu(0, 0);
        }
        static private data_result_alu ALU_BITE(byte cmd, byte
Nbit, byte contents_register)
        {
            switch (cmd)
            {
                case 0: //BCF
                    return new
data_result_alu((byte)(contents_register & ~(1 << Nbit)), 0);

                case 1: //BSF
                    return new
data_result_alu((byte)(contents_register | (1 << Nbit)), 0);

                case 2: //BTFSC
                    return new
data_result_alu(contents_register, 0, (((contents_register >>
Nbit) & 1) == 0) ? true : false);

                case 3: //BTFSS
                    return new
```

```
data_result_alu(contents_register, 0, (((contents_register >>
Nbit) & 1) == 1) ? true : false);
            }
            return new data_result_alu(0, 0);
        }
        static private byte Create_Status(int
result_operation, int low_order = 0)
        {
            byte Status = 0;

            if (result_operation == 0) //Z
                Status = (byte)(Status | 4);

            if (low_order < 0 || low_order > 255) //DC
                Status = (byte)(Status | 2);

            if (result_operation < 0 || result_operation >
15)//C
                Status = (byte)(Status | 1);
            return Status;
        }//составить новый статус
        static private ushort Adde_MUX(PIC16f887 mic, byte
addr)
        {
            if (addr == 5) {
                flg_RBIF = false;
                mic.latch_RBIF = 0;
            }
            if (addr == 0)
                return (ushort)(mic.Read_RAM(4) |
((mic.Read_RAM(3) & 128) << 1)); // косвенная
            return (ushort)(addr | ((mic.Read_RAM(3) & 96) <<
2)); // прямая
```

```
        }//адресация
        static public void Input_values_from_buttons(PIC16f887
mic, int nub_pin, int mode)
        {
            mutexObj.WaitOne();
            Events.Add(new Event(nub_pin, mode));
            mutexObj.ReleaseMutex();
        }//добавить событие на входе
        static public int Output_pin_digital_request(PIC16f887
mic, int port, int pin)
        {
            int TRIS = mic.Read_RAM(port + 133);
            if (((TRIS >> pin) & 1) == 0 || (port == 5 && pin
> 3))
                return mic.ports[port * 8 + pin];
            return 2;
        }
        static public int
Find_out_the_register_status(PIC16f887 mic, int addres)
        {
            switch (addres)
            {
                case 512:
                    return mic.WDT;
                case 513:
                    return mic.Wreg;
                case 514:
                    return mic.Prescaler_8b;
                case 515:
                    return mic.Prescaler_16b | 65536;
                default:
                    if(addres >= 0 && addres < 512)
                        return mic.Read_RAM(addres);
                    else
```

```
                    return -1;
            }
        }  //состояние по адресу
        static public string
Find_out_the_register_name(PIC16f887 mic, int addres)
        {
            switch (addres)
            {
                case 512:
                    return "WDT";
                case 513:
                    return "WREG";
                case 514:
                    return "PRESCALER_8B";
                case 515:
                    return "PRESCALER_16B";
                default:
                    if(addres >= 0 && addres < 512) {
                    string name =
mic.Determine_register_name(addres);
                    if (name != "")
                        return name;
                    else
                        return "0x" + Convert.ToString(addres,
16).PadLeft(3, '0');
                    }
                    break;
            }
            return "";
        } //название по адресу
        static public int
Find_out_the_register_addres(PIC16f887 mic, string name)
        {
            name = name.ToUpper();
```

```
            if (!Regex.IsMatch(name, "^\\s*0X([0-
9]|A|B|C|D|E|F)+\\s*$") &&
Array.IndexOf(mic.Get_adrres_registers(),name) == -1)
                return -1;
            switch (name)
            {
                case "WDT":
                    return 512;
                case "WREG":
                    return 513;
                case "PRESCALER_8B":
                    return 514;
                case "PRESCALER_16B":
                    return 515;
                default:
                    if(mic.Key_designations(name) ==
"name_registers")
                        return
Convert.ToInt32(mic.Take_addres_registers(name), 16);
                    return Convert.ToInt32(name.Substring(2),
16);
            }
        } //адрес по названию
        static private bool Check_inclusion(PIC16f887 mic)
        {
            if (flg_RBIF) {
                mic.Write_RAM(11, (byte)(mic.Read_RAM(11) |
1));
                mic.latch_RBIF = 1;
            }
            int Inicon = mic.Read_RAM(11);

            bool PEI = (Inicon & 64) == 64; //переферийные
устройства
```

```
            if (PEI){
                int PIR1 = mic.Read_RAM(12), PIE1 =
mic.Read_RAM(140);
                int PIR2 = mic.Read_RAM(13), PIE2 =
mic.Read_RAM(141);
                PEI = ((PIR1 & PIE1) != 0) | ((PIR2 & PIE2) !=
0);
            }

            bool T0I = (Inicon & 36) == 36; //tmr0
            bool INT = (Inicon & 18) == 18; //внешние прерывания
            bool RBI = (Inicon & 9) == 9; //любое внешние
прерывания

            return PEI | T0I | INT | RBI;
        }
        static private bool Go_to_inclusion(PIC16f887 mic,
bool INTER)
        {
            if (INTER && ((mic.Read_RAM(11) & 128) == 128))
            {
                mic.Write_RAM(11, (byte)(mic.Read_RAM(11) &
127));
                ushort PCL = mic.Read_RAM(2);
                ushort PCLATH = mic.Read_RAM(10);
                ushort PC = (ushort)(((PCLATH & 31) << 8) |
PCL);
                mic.Push(PC);

                PC = 4;
                mic.Write_RAM(2, (byte)PC);
                mic.Write_RAM(10, (byte)(PC >> 8));
                Time_of_running += mic.Time_of_1_cycle(2);
                return true;
```

```
            }
            else
                return false;
        }
    }
}
```

Файл mic.cs

```
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Diagnostics.Eventing.Reader;
using System.Drawing;
using System.Drawing.Design;
using System.IO;
using System.Linq;
using System.Runtime.Intrinsics.Arm;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Input;
using static System.Net.Mime.MediaTypeNames;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.TaskbarCl
ock;

namespace dip
{
    internal class PIC16f887
    {
        ushort numb_cmd;
        public byte WDT;
```

```
        public byte Prescaler_8b;
        public ushort Prescaler_16b;
        byte Prescaler_TMR1;
        public byte Wreg;
        ushort[] Stack;
        int stack_pointer;
        ushort[] ROM;
        byte[] RAM;
        decimal LFINTOSC = 31000;
        decimal HFINTOSC = 8000000;
        decimal time_for_WDT;
        bool T0CKI_previous;
        public int latch_RBIF = 0;
        public int TMR0_delay = 0;

        Dictionary<string, ushort> jump_label;
        public string[] pins = new string[] {
        "RA0", "RA1", "RA2", "RA3", "RA4", "RA5", "RA6",
"RA7",
        "RB0", "RB1", "RB2", "RB3", "RB4", "RB5", "RB6",
"RB7",
        "RC0", "RC1", "RC2", "RC3", "RC4", "RC5", "RC6",
"RC7",
        "RD0", "RD1", "RD2", "RD3", "RD4", "RD5", "RD6",
"RD7",
        "RE0", "RE1", "RE2", "RE3"};

        public string[] ports_name = new string[] { "PORTA",
"PORTB", "PORTC", "PORTD", "PORTE" };
        public int[] ports;
        public Dictionary<string, string> name_registers = new
Dictionary<string, string>() {
            //банк памяти 1
            /*Indirect addr.*/ { "TMR0",   "001"},{ "PCL",
```

```
"002"},{ "STATUS", "003"},
            { "FSR",    "004"},{ "PORTA",  "005"},{ "PORTB",
"006"},{ "PORTC",  "007"},
            { "PORTD",  "008"},{ "PORTE",  "009"},{ "PCLATH",
"00A"},{ "INTCON", "00B"},
            { "PIR1",   "00C"},{ "PIR2",   "00D"},{ "TMR1L",
"00E"},{ "TMR1H",  "00F"},

            { "T1CON",  "010"},{ "TMR2",   "011"},{ "T2CON",
"012"},{ "SSPBUF", "013"},
            { "SSPCON", "014"},{ "CCPR1L", "015"},{ "CCPR1H",
"016"},{ "CCP1CON","017"},
            { "RCSTA",  "018"},{ "TXREG",  "019"},{ "RCREG",
"01A"},{ "CCPR2",  "01B"},
            { "CCPR2H", "01C"},{ "CCP2CON","01D"},{ "ADRESH",
"01E"},{ "ADCON0", "01F"},

            //банк памяти 2
            /*Indirect addr.*/ { "OPTION_REG","081"},/*   PCL
*/ /*    STATUS    */
            /*      FSR     */ { "TRISA",  "085"},{ "TRISB",
"086"},{ "TRISC",  "087"},
            { "TRISD",  "088"},{ "TRISE",  "089"},/*    PCLATH
*/ /*   INTCON     */
            { "PIE1",   "08C"},{ "PIE2",   "08D"},{ "PCON",
"08E"},{ "OSCCON", "08F"},

            { "OSCTUNE","090"},{ "SSPCON2","091"},{ "PR2",
"092"},{ "SSPADD", "093"},
            { "SSPSTAT","094"},{ "WPUB"   ,"095"},{ "IOCB",
"096"},{ "VRCON",  "097"},
            { "TXSTA",  "098"},{ "SPBRG",  "099"},{ "SPBRGH",
"09A"},{ "PWM1CON","09B"},
            { "ECCPAS", "09C"},{ "PSTRCON","09D"},{ "ADRESL",
```

```
"09E"},{ "ADCON1", "09F"},

            //банк памяти 3
            /*Indirect addr.*/ /*     TMR0     */ /*     PCL
*/ /*    STATUS    */
            /*     FSR      */ { "WDTCON", "105"},/*     PORTB
*/ { "CM1CON0","107"},
            { "CM2CON0","108"},{ "CM2CON1","109"},/*    PCLATH
*/ /*    INTCON    */
            { "EEDATA", "10C"},{ "EEADR",  "10D"},{ "EEDATH",
"10E"},{ "EEADRH", "10F"},

            //банк памяти 4
            /*Indirect addr.*/ /*     TMR0     */ /*      PCL
*/ /*    STATUS    */
            /*     FSR      */ { "SRCON",  "185"},/*     TRISB
*/{ "BAUDCTL", "187"},
            { "ANSEL",  "188"},{ "ANSELH", "189"},/*    PCLATH
*/ /*    INTCON    */
            { "EECON1", "18C"},{ "EECON2", "18D"} /*
Reserved             */
        };
        Dictionary<string, string> name_bits = new
Dictionary<string, string>() {
            { "W",     "0"},
            //STATUS
            { "IRP",   "7"},{ "RP1",   "6"},{ "RP0",   "5"},
                              { "Z",     "2"},{ "DC",    "1"},{
"C",     "0"},
            //PORTA
            { "RA7",   "7"},{ "RA6",   "6"},{ "RA5",   "5"},{
"RA4",   "4"},
            { "RA3",   "3"},{ "RA2",   "2"},{ "RA1",   "1"},{
"RA0",   "0"},
```

```
            //PORTB
            { "RB7",   "7"},{ "RB6",   "6"},{ "RB5",   "5"},{
"RB4",   "4"},
            { "RB3",   "3"},{ "RB2",   "2"},{ "RB1",   "1"},{
"RB0",   "0"},

            //PORTC
            { "RC7",   "7"},{ "RC6",   "6"},{ "RC5",   "5"},{
"RC4",   "4"},
            { "RC3",   "3"},{ "RC2",   "2"},{ "RC1",   "1"},{
"RC0",   "0"},

            //PORTD
            { "RD7",   "7"},{ "RD6",   "6"},{ "RD5",   "5"},{
"RD4",   "4"},
            { "RD3",   "3"},{ "RD2",   "2"},{ "RD1",   "1"},{
"RD0",   "0"},

            //PORTE
            { "RE3",   "3"},{ "RE2",   "2"},{ "RE1",   "1"},{
"RE0",   "0"},

            //INTCON
            { "GIE",   "7"},{ "PEIE",  "6"},{ "T0IE",  "5"},{
"INTE",  "4"},
            { "RBIE",  "3"},{ "T0IF",  "2"},{ "INTF",  "1"},{
"RBIF",  "0"},

            //PIR1
                          { "ADIF",  "6"},{ "RCIF",  "5"},{
"TXIF",  "4"},
            { "SSPIF", "3"},{ "CCP1IF","2"},{ "TMR2IF","1"},{
"TMR1IF","0"},
```

```
            //PIR2
            { "OSFIF", "7"},{ "C2IF",  "6"},{ "C1IF",  "5"},{
"EEIF",  "4"},
            { "BCLIF", "3"},{ "ULPWUIF","2"},              {
"CCP2IF","0"},

            //T1CON
            { "T1GINV", "7"},{ "TMR1GE","6"},{
"T1CKPS1","5"},{ "T1CKPS0","4"},
            { "T1OSCEN","3"},{ "T1SYNC","2"},{ "TMR1CS",
"1"},{ "TMR1ON", "0"},

            //T2CON
                              { "TOUTPS3","6"},{
"TOUTPS2","5"},{ "TOUTPS1","4"},
            { "TOUTPS0", "3"},{ "TMR2ON", "2"},{
"T2CKPS1","1"},{ "T2CKPS0","0"},

            //SSPCON
            { "WCOL",    "7"},{ "SSPOV",  "6"},{ "SSPEN",
"5"},{ "CKP",    "4"},
            { "SSPM3",   "3"},{ "SSPM2",  "2"},{ "SSPM1",
"1"},{ "SSPM0",  "0"},

            //CCP1CON
            { "P1M1",    "5"},{ "P1M0",  "5"},{ "DC1B1",
"5"},{ "DC1B0",  "4"},
            { "CCP1M3",  "3"},{ "CCP1M2","2"},{ "CCP1M1",
"1"},{ "CCP1M0", "0"},

            //RCSTA
            { "SPEN",    "7"},{ "RX9",   "6"},{ "SREN",
"5"},{ "CREN",   "4"},
            { "ADDEN",   "3"},{ "FERR",  "2"},{ "OERR",
"1"},{ "RX9D",   "0"},
```

```
            //CCP2CON
                                                { "DC2B1",
"5"},{ "DC2B0",  "4"},
            { "CCP2M3",  "3"},{ "CCP2M2","2"},{ "CCP2M1",
"1"},{ "CCP2M0", "0"},

            //ADCON0
            { "ADCS1",   "7"},{ "ADCS0", "6"},{ "CHS3",
"5"},{ "CHS2",   "4"},
            { "CHS1",    "3"},{ "CHS0",  "2"},{"GO/DONE",
"1"},{ "ADON",   "0"},

//*************************************************************
******

            //OPTION_REG
            { "RBPU",    "7"},{ "INTEDG","6"},{ "T0CS",
"5"},{ "T0SE",   "4"},
            { "PSA",     "3"},{ "PS2",   "2"},{ "PS1",
"1"},{ "PS0",    "0"},

            //TRISA
            { "TRISA7",  "7"},{ "TRISA6","6"},{ "TRISA5",
"5"},{ "TRISA4", "4"},
            { "TRISA3",  "3"},{ "TRISA2","2"},{ "TRISA1",
"1"},{ "TRISA0", "0"},

            //TRISB
            { "TRISB7",  "7"},{ "TRISB6","6"},{ "TRISB5",
"5"},{ "TRISB4", "4"},
            { "TRISB3",  "3"},{ "TRISB2","2"},{ "TRISB1",
"1"},{ "TRISB0", "0"},

            //TRISC
```

```
            { "TRISC7",  "7"},{ "TRISC6","6"},{ "TRISC5",
"5"},{ "TRISC4", "4"},
            { "TRISC3",  "3"},{ "TRISC2","2"},{ "TRISC1",
"1"},{ "TRISC0", "0"},

            //TRISD
            { "TRISD7",  "7"},{ "TRISD6","6"},{ "TRISD5",
"5"},{ "TRISD4", "4"},
            { "TRISD3",  "3"},{ "TRISD2","2"},{ "TRISD1",
"1"},{ "TRISD0", "0"},

            //TRISE
            { "TRISE3",  "3"},{ "TRISE2","2"},{ "TRISE1",
"1"},{ "TRISE0", "0"},


            //PIE1
                              { "ADIE",   "6"},{ "RCIE",
"5"},{ "TXIE",   "4"},
            { "SSPIE",   "3"},{ "CCP1IE", "2"},{ "TMR2IE",
"1"},{ "TMR1IE", "0"},

            //PIE2
            { "OSFIE",   "7"},{ "C2IE",   "6"},{ "C1IE",
"5"},{ "EEIE",   "4"},
            { "BCLIE",   "3"},{ "ULPWUIE","2"},
{ "CCP2IE", "0"},

            //PCON
                                                { "ULPWUE",
"5"},{ "SBOREN", "4"},
                                                { "POR",
"1"},{ "BOR",    "0"},

            //OSCCON
```

```
                                { "IRCF2",  "6"},{ "IRCF1",
"5"},{ "IRCF0",  "4"},
            { "OSTS",    "3"},{ "HTS",    "2"},{ "LTS",
"1"},{ "SCS",    "0"},

            //OSCTUNE

{ "TUN4",   "4"},
            { "TUN3",    "3"},{ "TUN2",   "2"},{ "TUN1",
"1"},{ "TUN0",   "0"},

            //SSPCON2
            { "GCEN",    "7"},{ "ACKSTAT","6"},{ "ACKDT",
"5"},{ "ACKEN",  "4"},
            { "RCEN",    "3"},{ "PEN",    "2"},{ "RSEN",
"1"},{ "SEN",    "0"},

            //SSPMSK
            { "MSK7",    "7"},{ "MSK6",   "6"},{ "MSK5",
"5"},{ "MSK4",   "4"},
            { "MSK3",    "3"},{ "MSK2",   "2"},{ "MSK1",
"1"},{ "MSK0",   "0"},

            //SSPSTAT
            { "SMP",     "7"},{ "CKE",    "6"},{ "D/A",
"5"},{ "P",      "4"},
            { "S",       "3"},{ "R/W",    "2"},{ "UA",
"1"},{ "BF",     "0"},

            //WPUB
            { "WPUB7",   "7"},{ "WPUB6",  "6"},{ "WPUB5",
"5"},{ "WPUB4",  "4"},
            { "WPUB3",   "3"},{ "WPUB2",  "2"},{ "WPUB1",
"1"},{ "WPUB0",  "0"},
```

```
            //IOCB
            { "IOCB7",   "7"},{ "IOCB6",  "6"},{ "IOCB5",
"5"},{ "IOCB4",  "4"},
            { "IOCB3",   "3"},{ "IOCB2",  "2"},{ "IOCB1",
"1"},{ "IOCB0",  "0"},

            //VRCON
            { "VREN",    "7"},{ "VROE",   "6"},{ "VRR",
"5"},{ "VRSS",   "4"},
            { "VR3",     "3"},{ "VR2",    "2"},{ "VR1",
"1"},{ "VR0",    "0"},

            //TXSTA
            { "CSRC",    "7"},{ "TX9",     "6"},{ "TXEN",
"5"},{ "SYNC",   "4"},
            { "SENDB",   "3"},{ "BRGH",    "2"},{ "TRMT",
"1"},{ "TX9D",   "0"},

            //SPBRG
            { "BRG7",    "7"},{ "BRG6",    "6"},{ "BRG5",
"5"},{ "BRG4",   "4"},
            { "BRG3",    "3"},{ "BRG2",    "2"},{ "BRG1",
"1"},{ "BRG0",   "0"},

            //SPBRGH
            { "BRG15",   "7"},{ "BRG14",   "6"},{ "BRG13",
"5"},{ "BRG12",  "4"},
            { "BRG11",   "3"},{ "BRG10",   "2"},{ "BRG9",
"1"},{ "BRG8",   "0"},

            //PWM1CON
            { "PRSEN",   "7"},{ "PDC6",    "6"},{ "PDC5",
"5"},{ "PDC4",   "4"},
            { "PDC3",    "3"},{ "PDC2",    "2"},{ "PDC1",
"1"},{ "PDC0",   "0"},
```

```
            //ECCPAS
            { "ECCPASE", "7"},{ "ECCPAS2", "6"},{ "ECCPAS1",
"5"},{ "ECCPAS0","4"},
            { "PSSAC1",  "3"},{ "PSSAC0",  "2"},{ "PSSBD1",
"1"},{ "PSSBD0", "0"},

            //PSTRCON

{ "STRSYNC", "4"},
            { "STRD",    "3"},{ "STRC",    "2"},{ "STRB",
"1"},{ "STRA",    "0"},

            //ADCON1
            { "ADFM",    "7"},                  { "VCFG1",
"5"},{ "VCFG0",   "4"},

//*************************************************************
******

            //WDTCON

{ "WDTPS3",  "4"},
            { "WDTPS2",  "3"},{ "WDTPS1",  "2"},{ "WDTPS0",
"1"},{ "SWDTEN",  "0"},

            //CM1CON0
            { "C1ON",    "7"},{ "C1OUT",   "6"},{ "C1OE",
"5"},{ "C1POL",   "4"},
                                { "C1R",     "2"},{ "C1CH1",
"1"},{ "C1CH0",   "0"},

            //CM2CON0
            { "C2ON",    "7"},{ "C2OUT",   "6"},{ "C2OE",
"5"},{ "C2POL",   "4"},
```

```
                                { "C2R",     "2"},{ "C2CH1",
"1"},{ "C2CH0",   "0"},

            //CM2CON1
            { "MC1OUT",  "7"},{ "MC2OUT",  "6"},{ "C1RSEL",
"5"},{ "C2RSEL",  "4"},
                                                { "T1GSS",
"1"},{ "C2SYNC",  "0"},

            //EEDAT
            { "EEDAT7",  "7"},{ "EEDAT6",  "6"},{ "EEDAT5",
"5"},{ "EEDAT4",  "4"},
            { "EEDAT3",  "3"},{ "EEDAT2",  "2"},{ "EEDAT1",
"1"},{ "EEDAT0",  "0"},

            //EEADR
            { "EEADR7",  "7"},{ "EEADR6",  "6"},{ "EEADR5",
"5"},{ "EEADR4",  "4"},
            { "EEADR3",  "3"},{ "EEADR2",  "2"},{ "EEADR1",
"1"},{ "EEADR0",  "0"},

            //EEADR
                                                { "EEDATH5",
"5"},{ "EEDATH4", "4"},
            { "EEDATH3", "3"},{ "EEDATH2", "2"},{ "EEDATH1",
"1"},{ "EEDATH0", "0"},

            //EEADRH

{ "EEADRH4", "4"},
            { "EEADRH3", "3"},{ "EEADRH2", "2"},{ "EEADRH1",
"1"},{ "EEADRH0", "0"},


//*************************************************************
```

```
******

            //SRCON
            { "SR1",     "7"},{ "SR0",     "6"},{ "C1SEN",
"5"},{ "C2REN",   "4"},
            { "PULSS",   "3"},{ "PULSR",   "2"},
{ "FVREN",   "0"},

            //BAUDCTL
            { "ABDOVF",  "7"},{ "RCIDL",   "6"},
{ "SCKP",    "4"},
            { "BRG16",   "3"},                { "WUE",
"1"},{ "ABDEN",   "0"},

            //ANSEL
            { "ANS7",    "7"},{ "ANS6",    "6"},{ "ANS5",
"5"},{ "ANS4",    "4"},
            { "ANS3",    "3"},{ "ANS2",    "2"},{ "ANS1",
"1"},{ "ANS0",    "0"},

            //ANSELH
                                                { "ANS13",
"5"},{ "ANS12",   "4"},
            { "ANS11",   "3"},{ "ANS10",   "2"},{ "ANS9",
"1"},{ "ANS8",    "0"},

            //EECON1
            { "EEPGD",   "7"},
            { "WRERR",   "3"},{ "WREN",    "2"},{ "WR",
"5"},{ "RD",      "0"}
        };
        Dictionary<string, string> comand = new
Dictionary<string, string>(){
            //Байтовые операции
```

```
{ "ADDWF",  "000111dfffffff"},
{ "ANDWF",  "000101dfffffff"},
{ "CLRF",   "0000011fffffff"},
{ "CLRW",   "0000010xxxxxxx"},
{ "COMF",   "001001dfffffff"},
{ "DECF",   "000011dfffffff"},
{ "DECFSZ", "001011dfffffff"},
{ "INCF",   "001010dfffffff"},
{ "INCFSZ", "001111dfffffff"},
{ "IORWF",  "000100dfffffff"},
{ "MOVF",   "001000dfffffff"},
{ "MOVWF",  "0000001fffffff"},
{ "NOP",    "0000000xx00000"},
{ "RLF",    "001101dfffffff"},
{ "RRF",    "001100dfffffff"},
{ "SUBWF",  "000010dfffffff"},
{ "SWAPF",  "001110dfffffff"},
{ "XORWF",  "000110dfffffff"},
//Битовые операции b и f
{ "BCF",    "0100bbbfffffff"},
{ "BSF",    "0101bbbfffffff"},
{ "BTFSC",  "0110bbbfffffff"},
{ "BTFSS",  "0111bbbfffffff"},
//Операции управления и работы с константами, только X
{ "ADDLW",  "11111xkkkkkkkk"},
{ "ANDLW",  "111001kkkkkkkk"},
{ "CALL",   "100kkkkkkkkkk"},
{ "CLRWDT", "00000001100100"},
{ "GOTO",   "101kkkkkkkkkk"},
{ "IORLW",  "111000kkkkkkkk"},
{ "MOVLW",  "1100xxkkkkkkkk"},
{ "RETFIE", "00000000001001"},
{ "RETLW",  "1101xxkkkkkkkk"},
{ "RETURN", "00000000001000"},
```

```
            { "SLEEP",  "00000001100011"},
            { "SUBLW",  "11110xkkkkkkkk"},
            { "XORLW",  "111010kkkkkkkk"}
        };
        public PIC16f887()
        {
            ports = new int[36];
            T0CKI_previous = false;
            time_for_WDT = 0;
            numb_cmd = 0;
            WDT = 0;
            Prescaler_8b = 0;
            Prescaler_16b = 0;
            Wreg = 0;
            Stack = new ushort[8];
            stack_pointer = 0;
            jump_label = new Dictionary<string, ushort>();
            ROM = new ushort[8191];

            RAM = new byte[512];
            RAM[3] = 24;    //STATUS
            RAM[131] = 24;
            RAM[259] = 24;
            RAM[387] = 24;
            RAM[129] = 255; //OPTION_REG
            RAM[385] = 24;
            RAM[133] = 255; //TRISA
            RAM[134] = 255; //TRISB
            RAM[390] = 255;
            RAM[135] = 255; //TRISC
            RAM[136] = 255; //TRISD
            RAM[137] = 15;  //TRISE
            RAM[142] = 16;  //PCON
            RAM[143] = 96;  //OSCCON
            RAM[146] = 255; //PR2
```

```
            RAM[149] = 255; //WPUB
            RAM[152] = 2;   //TXSTA
            RAM[157] = 1;   //PSTRCON
            RAM[157] = 8;   //WDTCON
            RAM[392] = 255; //ANSEL
            RAM[393] = 255;//ANSELH
        } // Power-on
        public PIC16f887(PIC16f887 mic)
        {
            ports = new int[36];
            T0CKI_previous = false;
            Prescaler_8b = 0;
            Prescaler_16b = 0;
            time_for_WDT = 0;
            jump_label = new Dictionary<string, ushort>();
            numb_cmd = 0;
            ROM = new ushort[8191];
            Stack = mic.Stack;
            stack_pointer = 0;
            Wreg = mic.Wreg;

            RAM = new byte[512];
            RAM[0] = mic.RAM[0];                        //INDF
            RAM[128] = mic.RAM[128];
            RAM[256] = mic.RAM[256];
            RAM[384] = mic.RAM[384];
            RAM[1] = mic.RAM[1];                        //TMR0
            RAM[257] = mic.RAM[257];
            RAM[3] = (byte)((mic.RAM[3] & 7) | 24);
//STATUS
            RAM[131] = (byte)((mic.RAM[131] & 7) | 24);
            RAM[259] = (byte)((mic.RAM[259] & 7) | 24);
            RAM[387] = (byte)((mic.RAM[387] & 7) | 24);
            RAM[4] = mic.RAM[4];                        //FSR
            RAM[132] = mic.RAM[132];
```

```
            RAM[260] = mic.RAM[260];
            RAM[388] = mic.RAM[388];
            RAM[5] = (byte)(mic.RAM[5] & 16);          //PORTA
            RAM[6] = mic.RAM[6];                       //PORTB
            RAM[262] = mic.RAM[262];
            RAM[7] = mic.RAM[7];                       //PORTC
            RAM[8] = mic.RAM[8];                       //PORTD
            RAM[9] = (byte)(mic.RAM[9] & 15);
//PORTE
            RAM[11] = (byte)(mic.RAM[11] & 1);
//INTCON
            RAM[139] = (byte)(mic.RAM[139] & 1);
            RAM[267] = (byte)(mic.RAM[267] & 1);
            RAM[395] = (byte)(mic.RAM[395] & 1);
            RAM[14] = mic.RAM[14];                     //TMR1L
            RAM[15] = mic.RAM[15];                     //TMR1H
            RAM[19] = mic.RAM[19];
//SSPBUF
            RAM[21] = mic.RAM[21];
//CCPR1L
            RAM[22] = mic.RAM[22];
//CCPR1H
            RAM[24] = (byte)(mic.RAM[24] & 1);         //RCSTA
            RAM[27] = mic.RAM[27];
//CCPR2L
            RAM[28] = mic.RAM[28];
//CCPR2H
            RAM[30] = mic.RAM[30];
//ADRESH
            RAM[129] = 255;
//OPTION_REG
            RAM[385] = 255;
//OPTION_REG
            RAM[133] = 255;                            //TRISA
            RAM[134] = 255;                            //TRISB
```

```
            RAM[135] = 255;                                 //TRISC
            RAM[136] = 255;                                 //TRISD
            RAM[137] = 15;                                  //TRISE
            RAM[142] = (byte)(16 | (mic.RAM[142] & 1));//PCON
            RAM[143] = 96;
//OSCCON                                  //обратить внимание
            RAM[146] = 255;                                 //PR2
            RAM[149] = 255;                                 //WPUB
            RAM[152] = 2;                                   //TXSTA
            RAM[152] = 1;
//PSTRCON
            RAM[158] = mic.RAM[158];
//ADRESL
            RAM[261] = 8;
//WDTCON
            RAM[391] = 64;
//BAUDCTL
            RAM[392] = mic.RAM[392];                        //ANSEL
            RAM[393] = mic.RAM[393];
//ANSELH
        } //Reset
        public bool Push(ushort data)
        {
            bool result = true;
            if (stack_pointer > 7)
            {
                stack_pointer = 0;
                result = false;
            }
            Stack[stack_pointer] = data;
            stack_pointer++;

            return result;
        }
```

```
        public ushort Pop()
        {
            stack_pointer--;
            if (stack_pointer < 0) stack_pointer = 7;
            ushort Value = Stack[stack_pointer];
            Stack[stack_pointer] = 0;
            return Value;
        }
        public void CLRWDT()
        {
            WDT = 0;
            if ((RAM[129] & 8) != 0)
                Prescaler_8b = 0;
        }
        public decimal Time_of_1_cycle(int cycle)
        {
            // расчет времени выполнения
            byte OSCCON = RAM[143];
            decimal frequency = 1;
            if ((OSCCON & 8) == 0) //Device is running from
the internal oscillator (HFINTOSC or LFINTOSC)
            {
                if ((OSCCON & 112) == 0)
                    frequency = LFINTOSC;
                else
                    frequency = HFINTOSC / (1 << (~(OSCCON >>
4) & 7));
            }

            //работа от внешнего источника
            decimal Time = 1 / frequency * 4;
            // работа WDT
            WDT_inc(cycle, Time);
            // работа Timer0
```

```
            if ((RAM[129] & 32) == 32) //T0CS = 1
            {
                if (((RAM[129] & 16) == 16 && T0CKI_previous
&& (RAM[5] & 16) == 0) || ((RAM[129] & 16) == 0 &&
!T0CKI_previous && (RAM[5]& 16) == 16)) // T0SE: 1 RA4: 1 > 0
/ T0SE: 0 RA4: 0 > 1
                    Timer0_inc(1);
            }
            else
                Timer0_inc(cycle); //T0CS = 0
            T0CKI_previous = (RAM[5] & 16) == 16;

            return Time * cycle;
        }
        private void WDT_inc(int cycle, decimal Time)
        {
            if ((RAM[261] & 1) == 1) // включение WDT
            {
                for (int i = 0; i < cycle; i++)
                {
                    time_for_WDT += Time;
                    while (time_for_WDT >= 1 / LFINTOSC)
                    {
                        int pt16 = Prescaler_16b;
                        Prescaler_16b += (ushort)(1 <<
Math.Max(0, 11 - (RAM[261] >> 1) & 15));
                        if (Prescaler_16b < pt16) // 16 битный
предделитель
                            if ((RAM[129] & 8) == 8)
                            {
                                if ((Prescaler_8b + (256 >>
(RAM[129] & 7))) >= 256)// 8 битный предделитель
                                    WDT++; //сделать переполнение
                                Prescaler_8b += (byte)(256 >>
```

```
(RAM[129] & 7));
                            }
                            else
                                WDT++;
                        time_for_WDT -= 1 / LFINTOSC;
                    }
                }
            }
        }
        private void Timer0_inc(int cycle)
        {
            while (cycle != 0)
            {   if (TMR0_delay > 0) {
                    TMR0_delay--;
                    cycle--;
                    continue;
                }
                if (((RAM[129] >> 3) & 1) == 0)
                {
                    if ((Prescaler_8b + (128 >> (RAM[129] &
7))) >= 256)// 8 битный предделитель
                        Write_RAM(1, ++RAM[1]);
                    Prescaler_8b += (byte)(256 >> (RAM[129] &
7));
                }
                else
                    Write_RAM(1, ++RAM[1]);
                if (RAM[1] == 0)
                {
                    Write_RAM(11, (byte)(RAM[11] | 4));
                    return;
                }
                cycle--;
            }
```

```
        }
        private void Timer1_inc(int cycle)
        {
            int T1CON = RAM[16];
            if ((T1CON & 1) == 0)
                return;

            if ((T1CON & 2) == 2)
            {
                Prescaler_TMR1++;
                if (1 << ((T1CON >> 4) & 3) == Prescaler_TMR1)
{
                    Prescaler_TMR1 = 0;
                    if (++RAM[14] == 0)
                        RAM[15]++;
                }
            }

        }
        public bool Save_command(ushort cmd, int addr)
        {
            if (numb_cmd < 8192)
            {
                ROM[addr] = cmd;
                numb_cmd++;
                return true;
            }
            return false;
        }
        public void Write_RAM(int n, byte result, bool
Is_result = false)
        {
            switch (n)
            {
                case 0: //INDF
```

```
case 128:
case 256:
case 384:
    Write_RAM(RAM[2], result);
    break;

case 1: //TMR0
case 257:
    RAM[1] = result;
    RAM[257] = result;
    if (Is_result) {
        if ((RAM[129] & 8) == 0)
            Prescaler_8b = 0;
        TMR0_delay = 2;
    }
    break;

case 2: //PCL
case 130:
case 258:
case 386:
    RAM[2] = result;
    RAM[130] = result;
    RAM[258] = result;
    RAM[386] = result;
    break;

case 3: //STATUS
case 131:
case 259:
case 387:
    if (Is_result)
        result = (byte)(result | 24);
    RAM[3] = result;
    RAM[131] = result;
```

```
                    RAM[259] = result;
                    RAM[387] = result;
                    break;

                case 4: //FSR
                case 132:
                case 260:
                case 388:
                    RAM[4] = result;
                    RAM[132] = result;
                    RAM[260] = result;
                    RAM[388] = result;
                    break;

                case 5: //PORTA
                    if (Is_result)
                    {
                        int mask = (RAM[392] & 15) |
((RAM[392] & 16) >> 1);
                        result = (byte)(result & ~RAM[133] &
~mask | RAM[5] & RAM[133] & mask);
                    }
                    RAM[5] = result;
                    break;

                case 6: //PORTB
                case 262:
                    if (Is_result) {
                        //         RB0/AN12
RB1/AN10                RB2/AN8                 RB3/AN9
RB4/AN11                RB5/AN13
                        int mask = ((RAM[393] & 16) >> 4) |
((RAM[393] & 4) >> 1) | ((RAM[393] & 1) << 2) | ((RAM[393] &
2) << 2) | ((RAM[393] & 8) << 1) | ((RAM[393] & 32));
                        result = (byte)(result & ~mask &
```

```
~RAM[134] | RAM[6] & mask & RAM[134]);
                    }
                    RAM[6] = result;
                    RAM[262] = result;
                    break;
                case 7: //PORTC
                    if (Is_result)
                        result = (byte)(result & ~RAM[135] |
RAM[7] & RAM[135]);
                    RAM[7] = result;
                    break;
                case 8: //PORTD
                    if (Is_result)
                        result = (byte)(result & ~RAM[136] |
RAM[8] & RAM[136]);
                    RAM[8] = result;
                    break;
                case 9: //PORTE
                    if (Is_result)
                    {
                        int mask = RAM[392] >> 5;
                        result = (byte)(result & ~RAM[137] &
~mask | RAM[9] & RAM[137] & ~mask);
                    }
                    RAM[9] = (byte)(result & 15);
                    break;

                case 10: //PCLATH
                case 138:
                case 266:
                case 394:
                    RAM[10] = (byte)(result & 31);
                    RAM[138] = (byte)(result & 31);
                    RAM[266] = (byte)(result & 31);
                    RAM[394] = (byte)(result & 31);
```

```
            break;

        case 11: //INTCON
        case 139:
        case 267:
        case 395:
            RAM[11] = (byte)(result | latch_RBIF);
            RAM[139] = (byte)(result | latch_RBIF);
            RAM[267] = (byte)(result | latch_RBIF);
            RAM[395] = (byte)(result | latch_RBIF);
            break;

        case 14: //TMR1L
            RAM[14] = result;
            if (Is_result)
                Prescaler_TMR1 = 0;
            break;

        case 15: //TMR1H
            RAM[15] = result;
            if (Is_result)
                Prescaler_TMR1 = 0;
            break;

        case 29: //CCP2CON
            RAM[29] = (byte)(result & 63);
            break;

        case 31: //ADCON0
            RAM[31] = (byte)(result & 223);
            break;

        case 129: //OPTION_REG
        case 385:
            RAM[129] = result;
```

```
            RAM[385] = result;
            break;

        case 134: //TRISB
        case 390:
            RAM[134] = result;
            RAM[390] = result;
            break;

        case 137: //TRISE
            RAM[137] = (byte)(result & 15);
            break;

        case 142: //PCON
            RAM[142] = (byte)(result & 51);
            break;

        case 143: //OSCCON
            RAM[143] = (byte)(result & 127);
            break;

        case 144: //OSCTUNE
            RAM[144] = (byte)(result & 31);
            break;

        case 157: //PSTRCON
            RAM[157] = (byte)(result & 31);
            break;

        case 159: //ADCON1
            RAM[159] = (byte)(result & 176);
            break;

        case 261: //WDTCON
            if((result & 30) >= 24)
```

```
                RAM[261] = (byte)(22 | result & 1);
            else
                RAM[261] = (byte)(result & 31);

            break;

        case 263: //CM1CON0
            RAM[263] = (byte)(result & 251);
            break;

        case 264: //CM2CON0
            RAM[264] = (byte)(result & 251);
            break;

        case 265: //CM2CON1
            RAM[265] = (byte)(result & 249);
            break;

        case 270: //EEDATH
            RAM[270] = (byte)(result & 63);
            break;

        case 271: //EEADRH
            RAM[271] = (byte)(result & 31);
            break;

        case 389: //SRCON
            RAM[389] = (byte)(result & 253);
            break;

        case 391: //BAUDCTL
            RAM[391] = (byte)(result & 219);
            break;

        case 396: //EECON1
```

```
            RAM[396] = (byte)(result & 15);
            break;

        case 397: //EECON2
            RAM[397] = (byte)(result & 0);
            break;

        default:
            if(n >= 112 & n <= 127)
            {
                RAM[n] = result;
                RAM[n + 128] = result;
                RAM[n + 256] = result;
                RAM[n + 384] = result;
                break;
            }
            if ( n >= 240 & n <= 255)
            {
                RAM[n - 128] = result;
                RAM[n] = result;
                RAM[n + 128] = result;
                RAM[n + 256] = result;
                break;
            }
            if (n >= 368 & n <= 383)
            {
                RAM[n - 256] = result;
                RAM[n - 128] = result;
                RAM[n] = result;
                RAM[n + 128] = result;
                break;
            }
            if (n >= 496 & n <= 511)
            {
                RAM[n - 384] = result;
```

```
                        RAM[n - 256] = result;
                        RAM[n - 128] = result;
                        RAM[n] = result;
                        break;
                    }
                    RAM[n] = result;
                    break;
                }
            }
            public byte Read_RAM(int n)
            {
                return RAM[n];
            }
            public ushort Read_ROM(int n)
            {
                return ROM[n];
            }
            public string[] Get_adrres_registers()
            {
                string[] result = new string[name_registers.Count
    + 4];
                int n = 0;
                foreach (var (key, val) in name_registers) {
                    result[n] = key;
                    n++;
                }
                result[name_registers.Count] = "WDT";
                result[name_registers.Count + 1] = "WREG";
                result[name_registers.Count + 2] = "PRESCALER_8B";
                result[name_registers.Count + 3] =
    "PRESCALER_16B";
                return result;
            }  //список всех регистров
            public string Determine_register_name(int value)
```

```
//название по адресу
        {
            foreach (var (key, val) in name_registers)
            {
                if (Convert.ToInt32(val, 16) == value)
                    return key;
            }
            return "";
        }
        public string Key_designations(string str)
        {
            if (name_registers.ContainsKey(str))
                return "name_registers";
            if (name_bits.ContainsKey(str))
                return "name_bits";
            if (comand.ContainsKey(str))
                return "comand";
                if (jump_label.ContainsKey(str))
                return "jump_label";
            return "absent";
        } //определить принадлежность к списку

        public ushort[] Stack_output() {

            return Stack;
        }
        public string Take_addres_registers(string str) //адрес
по названию
        {
            return name_registers[str];
        }
        public string Take_name_bits(string str)
        {
            return name_bits[str];
```

```
        }
        public string Take_comand(string str)
        {
            return comand[str];
        }
        public ushort Take_numb_cmd()
        {
            return numb_cmd;
        }
        public ushort Take_jump_label(string str)
        {
            return jump_label[str];
        }
        public bool TryAdd_jump_label(string cmd, ushort
numb_cmd)
        {
            if (jump_label.TryAdd(cmd, numb_cmd))
                return true;
            return false;
        }
    }
}
```

Приложение В

(Обязательное)

Учебный симулятор работы PIC микроконтроллера

Презентация

Учебный симулятор работы PIC микроконтроллера

Автор работы:
студент гр. 21ВВП2 Савицкий Макар Алексеевич
Научный руководитель:
к.т.н, доцент каф. ВТ Бычков Андрей Станиславович

Преимущества симуляторов микроконтроллеров

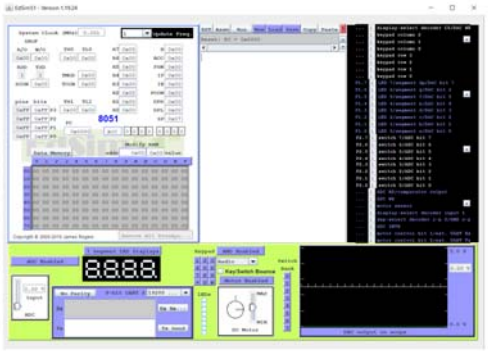
- Предотвращение повреждения оборудования
- Улучшение понимания работы микроконтроллера
- Упрощение и ускорение процесса отладки
- Безопасность виртуальной среды
- Экономия времени и ресурсов

Перед началом разработки необходимо:

- Проанализировать предметную область
- Определить возможности приложения
- Выбрать язык программирования и микроконтроллер

Анализ аналогичных решений

Симулятор МК8051



Плюсы:

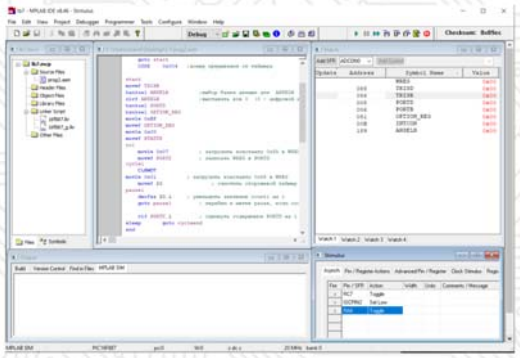
- Большой функционал периферии
- Возможность просмотреть регистры состояния
- Возможность просмотреть машинный код и состояние памяти

Минусы:

- Неудобная отладка программы

Анализ аналогичных решений

Приложение “MPLAB IDE”



Плюсы:

- Удобная отладка программы
- Просмотр параметров программы на временной шкале
- Возможность просмотра регистров

Минусы:

- Отсутствие полноценной работы с периферией

Приложение должно

- Иметь отзывчивый компилятор
- Обладать удобными форматами обработки команд
- Отображать все ЗУ элементов микроконтроллера
- В достаточной степени имитировать работу PIC микроконтроллера и его периферии

Выбор языка программирования

Выбор языка программирования

C#

Реализуемый PIC микроконтроллер

Архитектура приложения

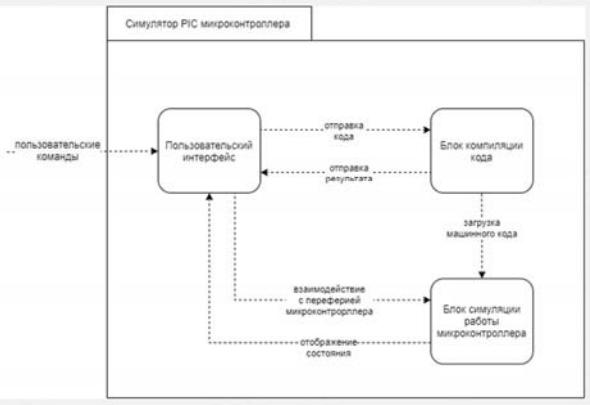
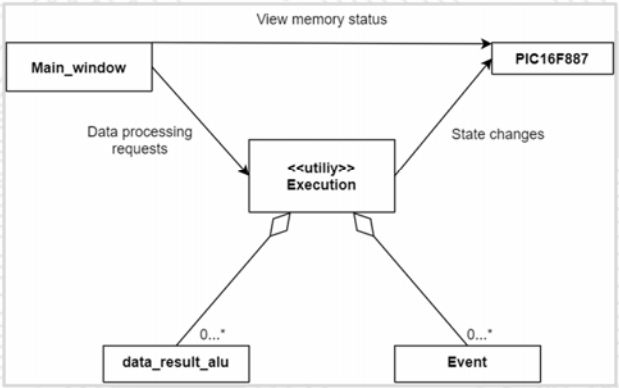
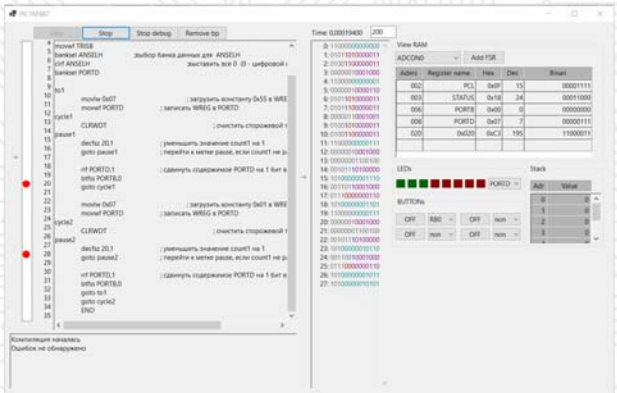


Диаграмма классов



Разработанное приложения



В приложении реализовано

- Удобный компилятор
- Подробный просмотр памяти устройства
- Удобные инструменты отладки
- Взаимодействие с периферией микроконтроллера

Рекомендации к дальнейшей разработке

- Оформить подсветку кода в программе
- Добавить нереализованную периферию
- Добавить отображение ЗУ нереализованных элементов микроконтроллера
- Сделать интерфейс в большей степени визуально привлекательным и стильным

Рекомендации к дальнейшей разработке

- Оформить подсветку кода в программе
- Добавить нереализованную периферию
- Добавить отображение ЗУ нереализованных элементов микроконтроллера
- Сделать интерфейс в большей степени визуально привлекательным и стильным