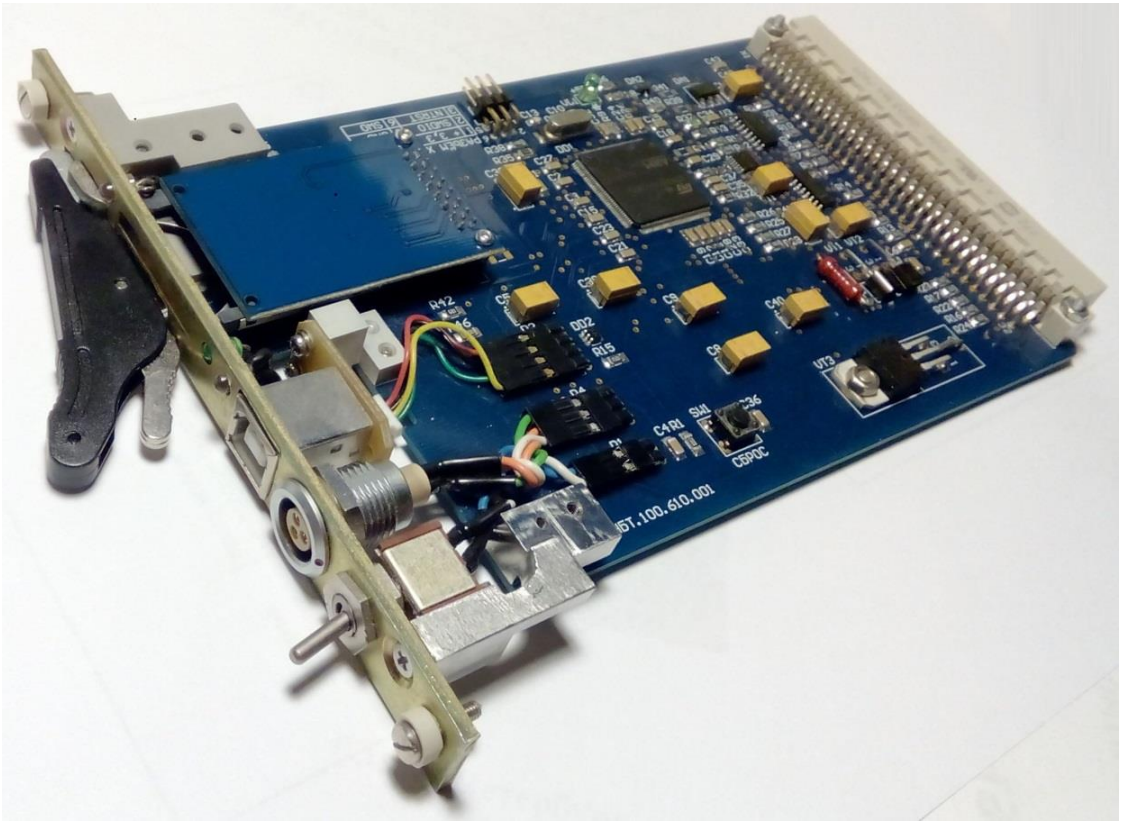
УНИВЕРСИТЕТ ИТМО

Ю.В. Федосов, А.С. Шубин

CORTEX M4. ЛАБОРАТОРНЫЙ ПРАКТИКУМ



Санкт-Петербург
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Ю.В. Федосов, А.С. Шубин

CORTEX M4. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 12.04.01 Приборостроение
в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования магистратуры

Санкт-Петербург
2020

Федосов Ю.В., Шубин А.С., Cortex M4. Лабораторный практикум.– СПб: Университет ИТМО, 2020. – 66 с.

Рецензент:
Афанасьев Максим Яковлевич, кандидат технических наук, доцент (квалификационная категория "ординарный доцент") факультета систем управления и робототехники, Университета ИТМО.

Пособие содержит пять лабораторных работ, предназначенных для практического освоения особенностей программирования и работы с ARM-процессорами, основанными на ядре Cortex M4. В ходе лабораторных работ используется программируемый контроллер STM32F407VGT6. Пособие предназначено для обеспечения усвоения студентами учебного материала по дисциплинам «Основы цифрового производства» и «Технологии приборостроения» по направлениям подготовки 15.03.04 «Автоматизация технологических процессов и производств» и 12.04.01 «Приборостроение» и направлено на формирование компетенции ПК-4 (способности организовывать и проводить технологическую подготовку производства элементов электронных средств, технических и киберфизических систем).

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Введение

В ходе данного курса Вы будете изучать особенности работы с микроконтроллерами с архитектурой ARM на примере контроллера STM32F407VGT6, серийно выпускаемого фирмой ST Microelectronics и базирующегося на ядре Cortex M4. Для работы с контроллером используется отладочная плата STM32F4 Discovery**.**

STM32F4 Discovery представляет собой высокопроизводительную отладочную плату и позволяет изучать возможности микроконтроллера STM32F407VGT6, а также разрабатывать собственные приложения. Отладочная плата имеет интегрированный отладчик ST-LINK/V2, два MEMS-устройства производства компании ST Microelectronics — цифровой измеритель ускорения (акселерометр) и цифровой микрофон, один аудио ЦАП с интегрированным драйвером громкоговорителя, работающим в классе D, светодиоды и кнопки, а также разъем USB OTG micro-AB. Также отметим, что устройство имеет поддержку в виде документации и большого количества бесплатных приложений, доступных на сайте [1].

На плате установлены:

• Микроконтроллер STM32F407VGT6, основывающийся на 32-разрядном ядре ARM Cortex-M4F, с 1 МБайт ЭСППЗУ и 192 кБайт ОЗУ;

• Встроенный отладчик ST-LINK/V2 с выбором режима работы, позволяющим использовать плату в качестве автономного программатора STLINK/ V2 (с разъемом SWD для программирования и отладки);

• Преобразователи питания, позволяющие осуществлять подачу напряжения питания на плату через шину USB или от внешнего источника питания 5 В;

• Средства питания внешних устройств, потребляющих 3,3 В, либо 5 В;

• Трёхосевой акселерометр с цифровым выходом LIS302DL;

• Всенаправленный цифровой микрофон MP45DT02;

• ЦАП CS43L22, с интегрированным усилителем класса D;

• Четыре пользовательских светодиода, LD3 (оранжевый), LD4 (зеленый), LD5 (красный) и LD6 (синий),

• Две кнопки: пользовательская и сброс (reset);

• USB OTG FS с разъемом micro-AB;

• Внешний разъем с выходом всех линий I/O корпуса LQFP100 для быстрого подключения к макетной плате и исследования сигналов.

• Средства контроля и индикации:

- светодиод LD1 (красный/зеленый) для отображения состояния подключения USB,

- светодиод LD2 (красный) для индикации наличия напряжения 3,3 В,

- два светодиода USB OTG — LD7 (зеленый) для индикации напряжения VBus и LD8 (красный), индицирующий перегрузку по току;

• Две кнопки: пользовательская и сброс (reset);

Также в комплект поставки платы входит кабель mini-USB, предназначенный для связи платы с ПК.

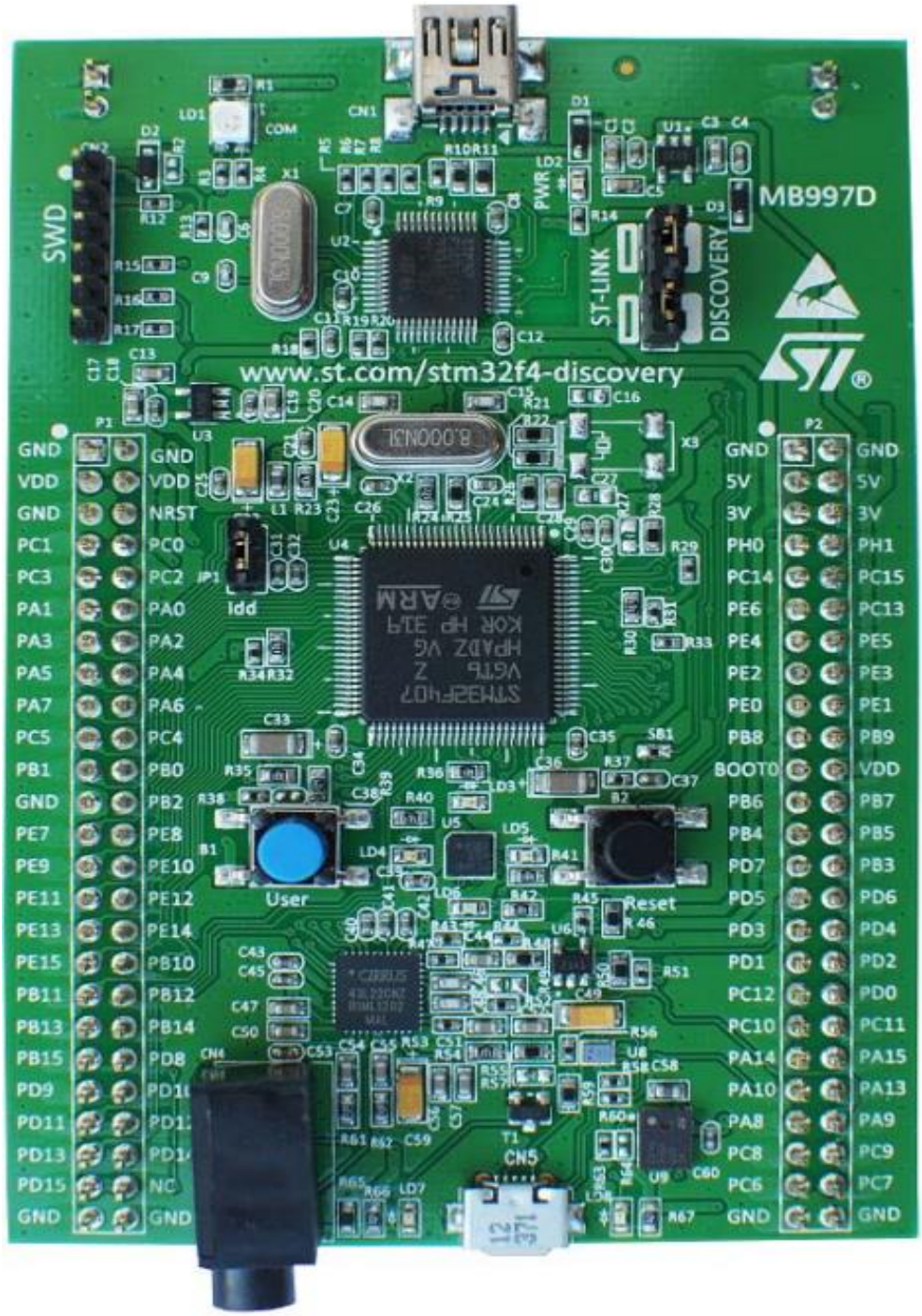


Рисунок 1 – Плата STM32F407 Discovery. Вид сверху

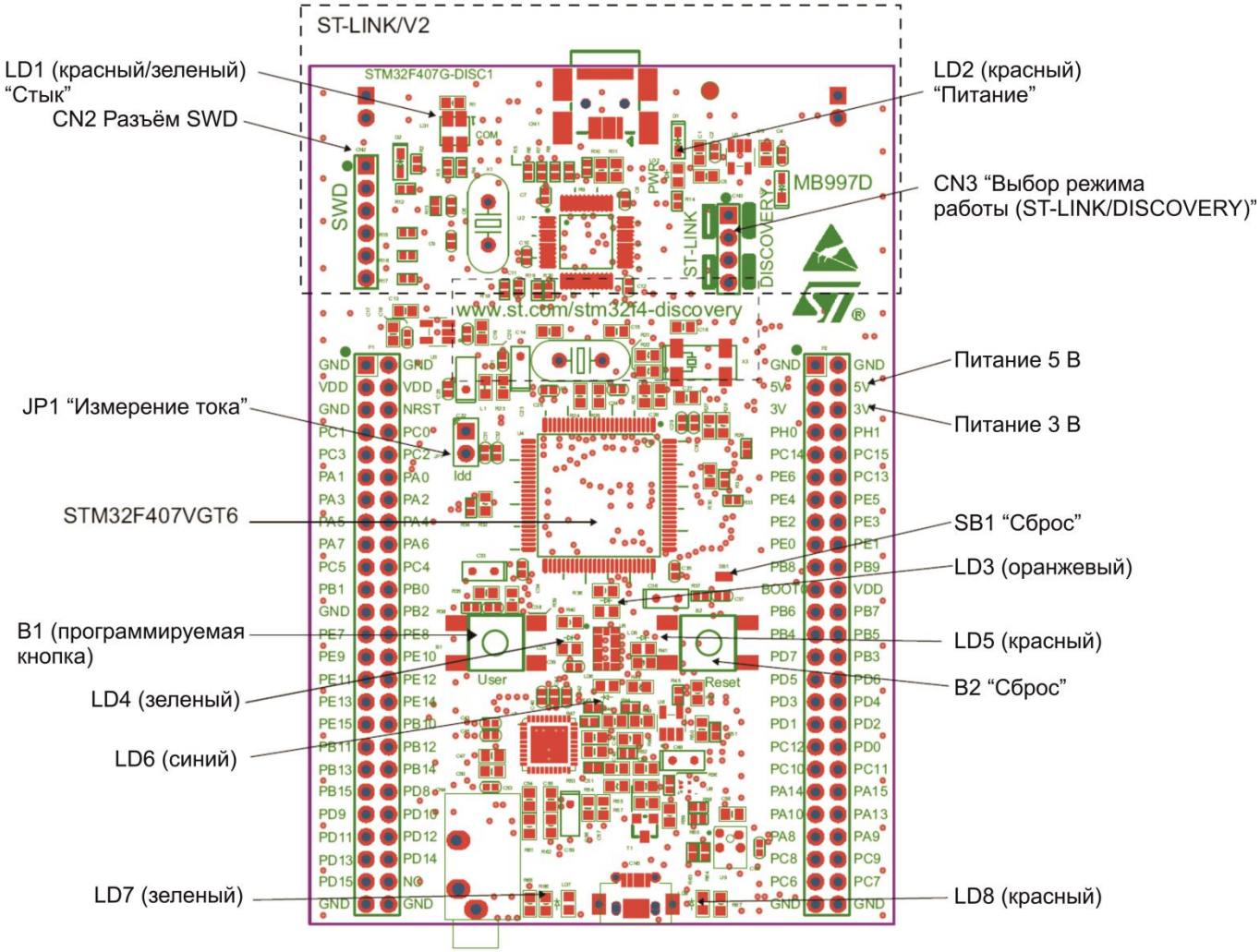


Рисунок 2 – Расположение основных элементов на плате STM32F407 Discovery

Системные требования к рабочему месту:

- ПК под управлением ОС Windows (2000, XP, Vista, 7);
- USB кабель типа A-Mini-B.
- Среда разработки с поддержкой ядра ARM.

Демонстрационное ПО

Плата поставляется потребителю с предварительно загруженным в ЭСППЗУ демонстрационным программным обеспечением. Демонстрационное программное обеспечение задействует акселерометр LIS302DL, при этом согласно данным, получаемым с него, осуществляется смена частоты прерывистого свечения светодиодов LD3...LD6. В случае, если плату подключить к ПК при помощи второго разъема и кабеля USB типа 'A — micro-B', то при движении платы происходит управление перемещением курсора по дисплею ПК.

Исходные коды последней версии демонстрационного программного обеспечения и сопутствующей документации можно загрузить с веб-сайта [2].

1 Лабораторная работа №1. Настройка среды разработки и создание проекта

1.2 Введение

Лабораторная работа №1 посвящена ознакомлению с ходом разработки и создания проекта для ARM – котроллера. Вы будете разрабатывать программы для микроконтроллера STM32 в среде CoIDE, которая основана на Eclipse. Программа CooCox CoIDE является свободным программным обеспечением, обладает необходимыми средствами разработки, а также содержит встроенный отладчик ST-Link, что позволяет подключать к ней отладочную плату и программировать контроллер.

В ходе лабораторной работы будет создан проект, а также произведены его настройки, настройки компьютера и определен необходимый минимум программного обеспечения для работы с платой STM32F5 Discovery. Также будут рассмотрены особенности установки драйверов и программ под ОС Windows.

1.3 Описание лабораторной работы

Цели работы

1. Научиться настраивать среду разработки для ARM – контроллеров.

Указания к выполнению работы

Лабораторная работа состоит из трёх частей. В ходе выполнения первой части производится создание и постройка проекта, затем настраиваются компилятор, отладчик и драйвер, после чего производится программирование контроллера.

Порядок выполнения работы

Часть 1:

1. Создайте проект в CoIDE;

2. Настройте проект в CoIDE.

Часть 2:

1. Настройте компилятор CoIDE;
2. Настройте отладчик CoIDE;
3. Настройте драйвер ST-Link.

Часть 3:

1. Напишите программу;
2. Скомпилируйте программу;
3. Загрузите программу в память котроллера.

Требования к отчету

Отчёт должен быть составлен в соответствии с требованиями, представленными в приложении А.

Отчет должен включать:

1. Титульный лист;
2. Цели работы;
3. Исходный код программы на языке C++;
4. Схема лабораторной установки;

1.4 Подготовка и описание программы CooCox CoIDE

Последнюю версию CooCox CoIDE можно скачать на официальном сайте CooCox [3]. Для скачивания необходимо зарегистрироваться (регистрация бесплатная). Затем загруженный файл необходимо запустить — таким образом будет установлена среда разработки. Устанавливать CoIDE рекомендуется в каталог без русских букв и пробелов, лучше всего — в каталог, предлагаемый по умолчанию.

При создании нового проекта предлагается выбор используемой микросхемы и библиотек. Возможен просмотр кратких характеристик каждой микросхемы. CoIDE автоматически создает всю структуру проекта, а также подключает все остальные необходимые для работы библиотеки. Каждая из них содержит несколько готовых примеров, которые можно использовать в проекте. Присутствует функция пополнения библиотек собственными примерами. При подключении новых библиотек к проекту учитываются все зависимости между ними.

Рабочая платформа рассматриваемой среды разработки – операционные системы Windows XP (необходим SP3), Vista (SP2), 7. Для установки CoIDE в Windows XP желательна версия Professional, так как в Home Edition не работает режим отладки. Способы решения этой проблемы описаны на форуме CooCox. Кроме этого, возможна работа с программой в среде Linux с помощью Wine. Однако функции отладки и записи в микроконтроллер будут недоступны.

CooCox CoIDE — среда разработки, которая помимо STM32 поддерживает ряд других семейств микроконтроллеров: Freescale, Holtek, NXP, Nuvoton, TI, Atmel SAM, Energy Micro и др. С каждой новой версией CoIDE список контроллеров постоянно пополняется.

После успешной установки CoIDE необходимо запустить программу.

1.5 Выполнение работы

Для того, чтобы создать проект в программе CooCox CoIDE, необходимо в главном меню программы выбрать Project → New project. Проект должен создаваться в папке, предлагаемой по умолчанию.

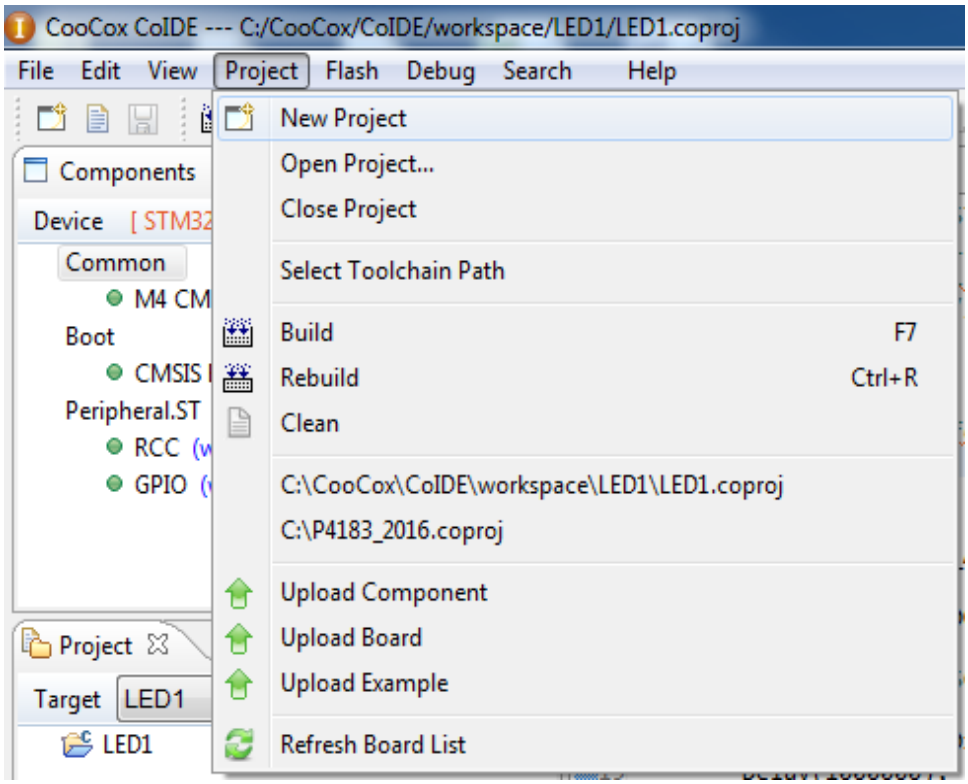


Рисунок 3 – Создание нового проекта

Далее в появившемся окне указывается имя проекта.

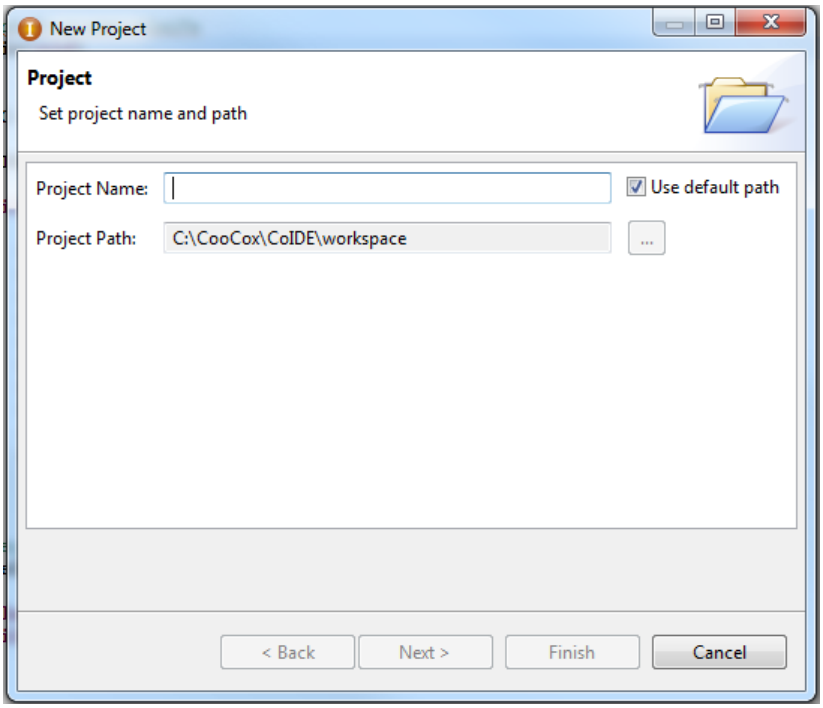


Рисунок 4 – Указание имени проекта

Затем в появившемся окне необходимо выбрать «Board», потом нажать «Next».

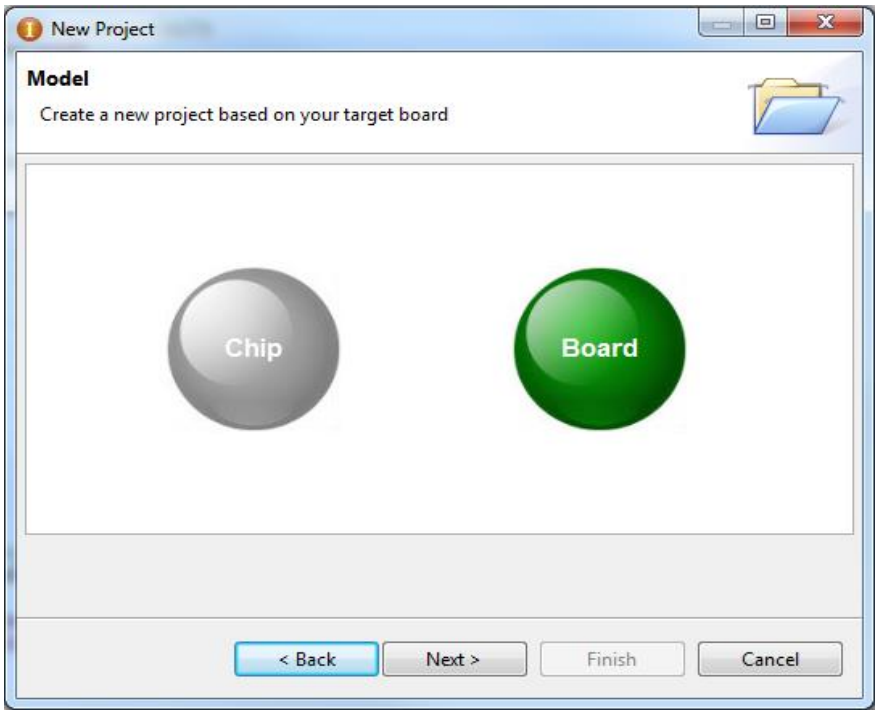


Рисунок 5 – Выбор платы

Из появившегося списка плат необходимо выбрать плату STM32F407 Discovery (stm32 → STM32F4x → stm32f4-Discovery), и далее нажать «Finish».

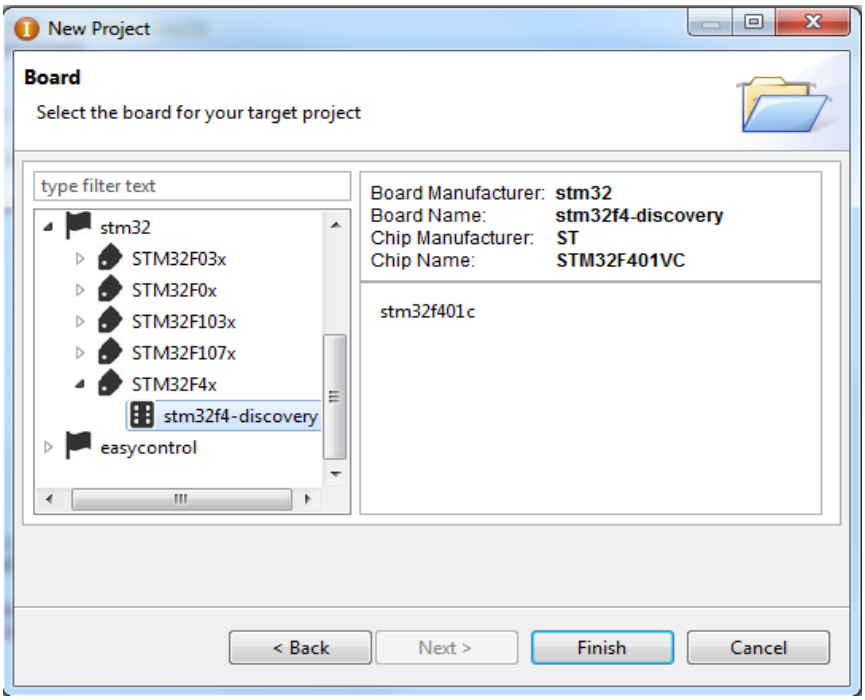


Рисунок 6 – Выбор отладочной платы

Справа, в окне **Help,** отображаются краткие характеристики каждого контроллера. После выбора нужного микроконтроллера требуется выбрать необходимые библиотеки для работы. В списке присутствуют библиотеки портов ввода-вывода GPIO, АЦП, UART, таймеры и т.д. Библиотеки постоянно добавляются и обновляются, также можно загрузить большое количество готовых библиотек (драйверов) для работы с разнообразными элементами или модулями (вкладка Drivers внизу окна), начиная от семисегментных индикаторов и заканчивая модулями камеры, GPS или популярных моделей радиомодулей.

В появившейся вкладке «Repository» необходимо перейти на вкладку «Peripherals» и отметить пункты «M4 CMSIS», «CMSIS BOOT», «RCC» и «GPIO». Можно заметить, что при выборе GPIO автоматически подключатся остальные необходимые для работы библиотеки (CMSIS Boot и RCC).

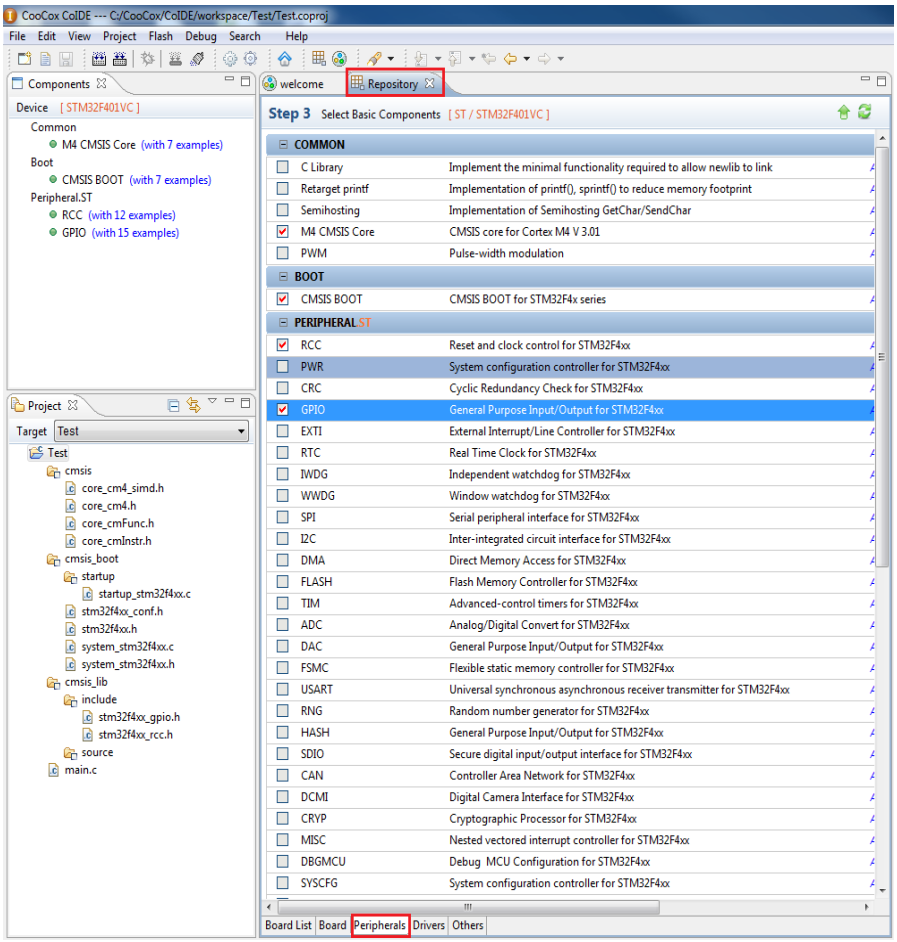


Рисунок 7 – Выбор библиотек

Далее в главном меню необходимо нажать «Edit», после чего выбрать пункт «Preferences» и появившемся окне перейти к пункту «Scalability» (C/C++ → Editor → Scalability), где необходимо ввести количество строк: 20000.

Отметим, что CoIDE позволяет загружать примеры прямо в текущий проект. На вкладке «Components» можно увидеть, что почти к каждой библиотеке есть примеры. Так, если нажать на **GPIO**, то можно увидеть 15 примеров.

При этом пользователь может добавлять свои примеры. В предлагаемых системой примерах уже присутствует пример программы для мигания светодиодом «GPIO_Blink». Чтобы добавить его в проект, необходимо нажать кнопку **Add.** Но в этом случае он будет добавлен как подключаемый файл, поэтому лучше просто скопировать весь код примера в файл main.c. При этом строку *void GPIO_Blink(void)* следует заменить на *int main(void)*.

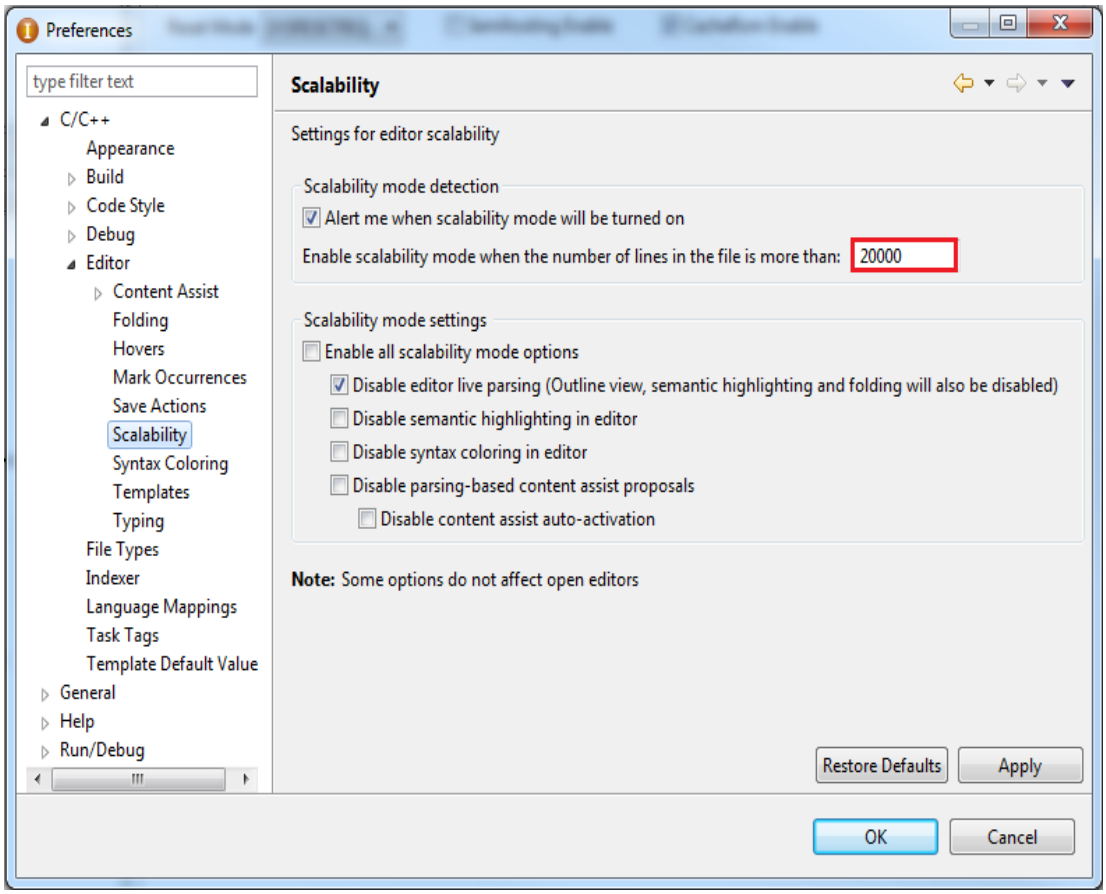


Рисунок 8 – Указание количества строк для редактора

Для того, чтобы скомпилировать программный код, среде нужен компилятор GCC. Чтобы скачать его, необходимо зайти на страницу GNU ARM Embedded Toolchain [4], далее нажать кнопку «Downloads» и пролистать появившуюся страницу до пункта «Windows 32-bit». Затем следует скачать последнюю версию компилятора, для чего нажать кнопку «Download». После необходимо установить компилятор, для этого необходимо запустить загруженный исполняемый файл, два раза щелкнув по нему. Затем необходимо выбрать язык, подтвердить своё согласие на установку и принять условия лицензионного соглашения. При установке следует принять предлагаемый по умолчанию адрес папки. Это гарантирует верную настройку компилятора. Отметим, что новая версия компилятора выходит приблизительно каждый месяц (как правило, одновременно с этим меняется дизайн сайта), поэтому процесс установки компилятора сложно формализовать. Подробное описание процесса установки можно найти по адресу [5]. Для того, чтобы среда разработки могла использовать компилятор, необходимо указать путь к нему. Для этого необходимо зайти в меню Project → Select Toolchain Path (рисунок 9). Затем в появившемся окне указывается путь к компилятору (рисунок 10).

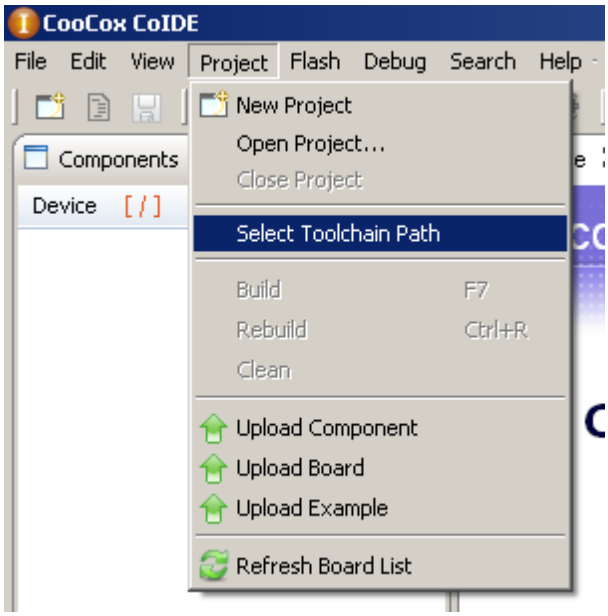


Рисунок 9 – Указание пути к компилятору

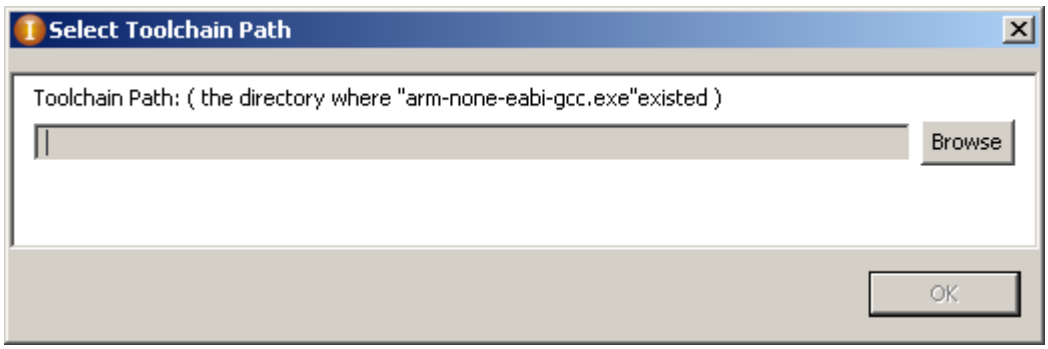


Рисунок 10 – Указание пути к компилятору

Далее необходимо удостовериться в том, что в качестве отладчика используется ST-Link. Для этого необходимо нажать на значок зубчатого колеса (рисунок 11) и в появившемся окне «Configuration» на вкладке «Debugger» выбрать «ST-Link» (рисунок 12).

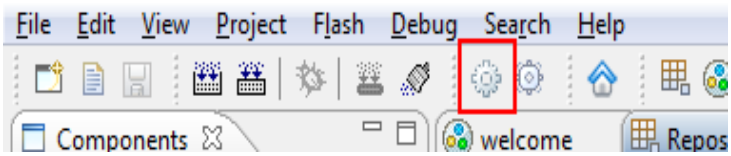


Рисунок 11 – (пояснение в тексте)

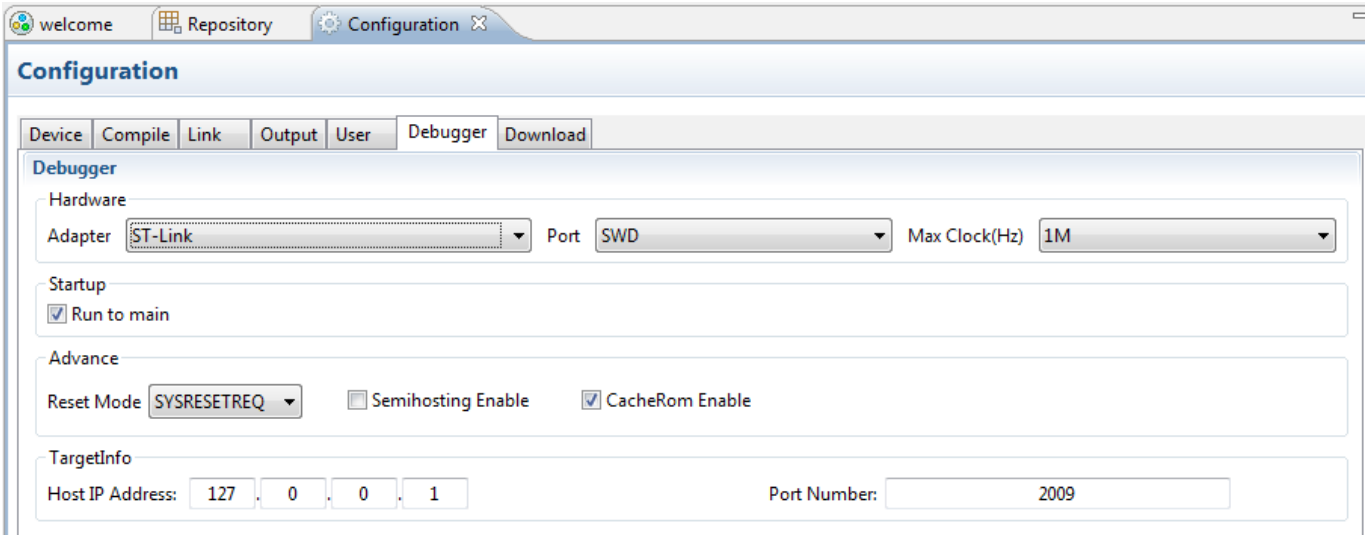


Рисунок 12 – Выбор отладчика

Для того, чтобы скомпилировать проект, необходимо нажать клавишу **F7**, либо выбрать в главном меню Project → Build.

Удостоверившись в том, что компиляция прошла успешно (сообщение BUILD SUCCESSFUL (рисунок 13)), необходимо загрузить программу в память контроллера.

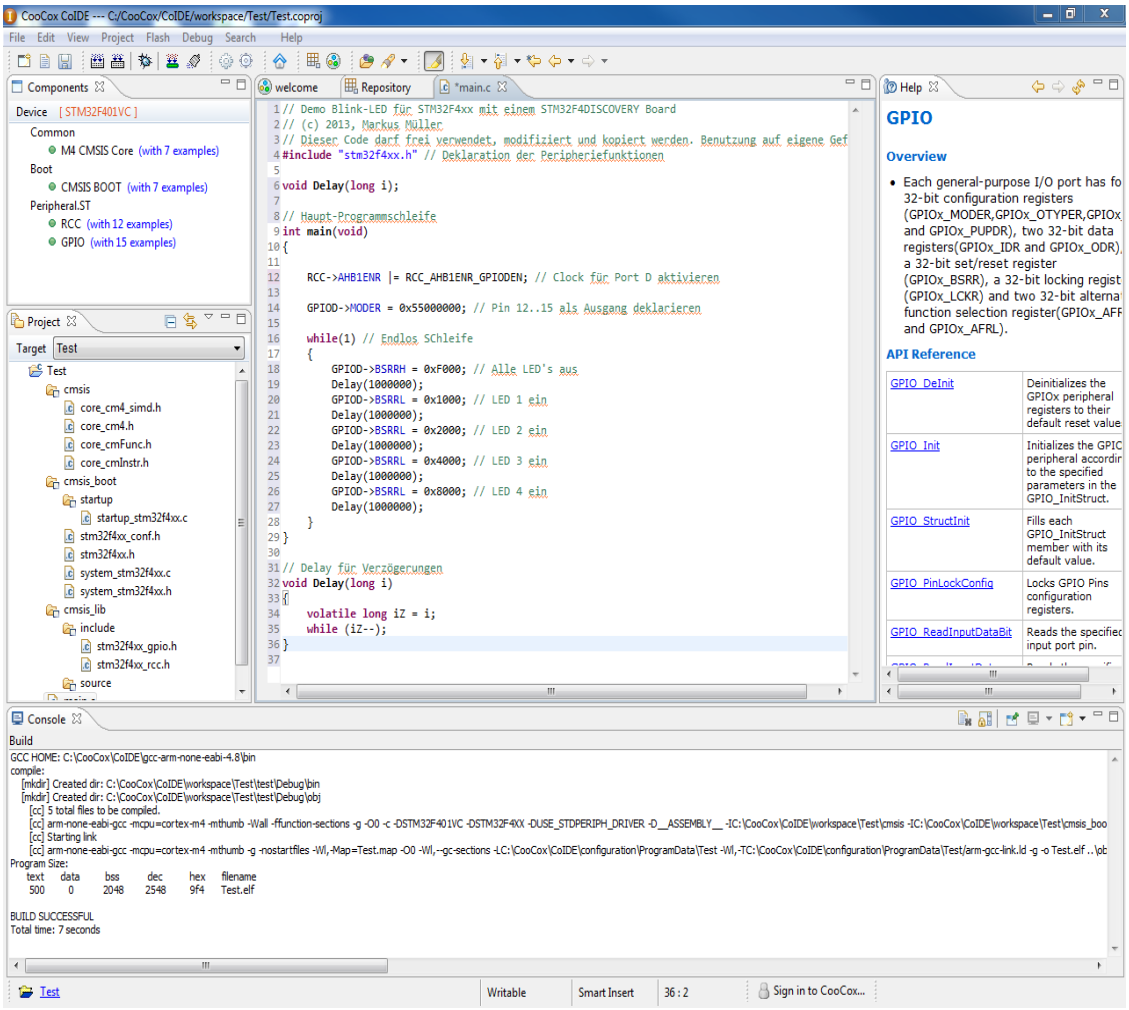


Рисунок 13 – Сообщение компилятора об успешной сборке

Перед тем, как программировать плату, необходимо установить драйвер ST-LINK. Его можно скачать с официальной страницы разработчика [6]. Например, для Windows XP это будет ST-LINK STSW-LINK004 (рисунок 14).

EMBEDDED SOFTWARE

DEVELOPMENT TOOL SOFTWARE

Part Number	Manufacturer	Description
STSW-LINK004	ST	STM32 ST-LINK utility
STSW-LINK007	ST	ST-LINK, ST-LINK/V2, ST-LINK/V2-1 firmware upgrade
STSW-LINK009	ST	ST-LINK, ST-LINK/V2, ST-LINK/V2-1 USB driver signed for Windows7, Windows8, Windows10

Рисунок 14 – Выбор драйвера

При установке драйвера следует установить его в папку "C:\CooCox\STM32 ST-LINK Utility".

В качестве тестового проекта будет создана программа, зажигающая и гасящая светодиоды PD3...PD6 с определённой частотой. Сами светодиоды подключены к контактам контроллера согласно следующей схеме:

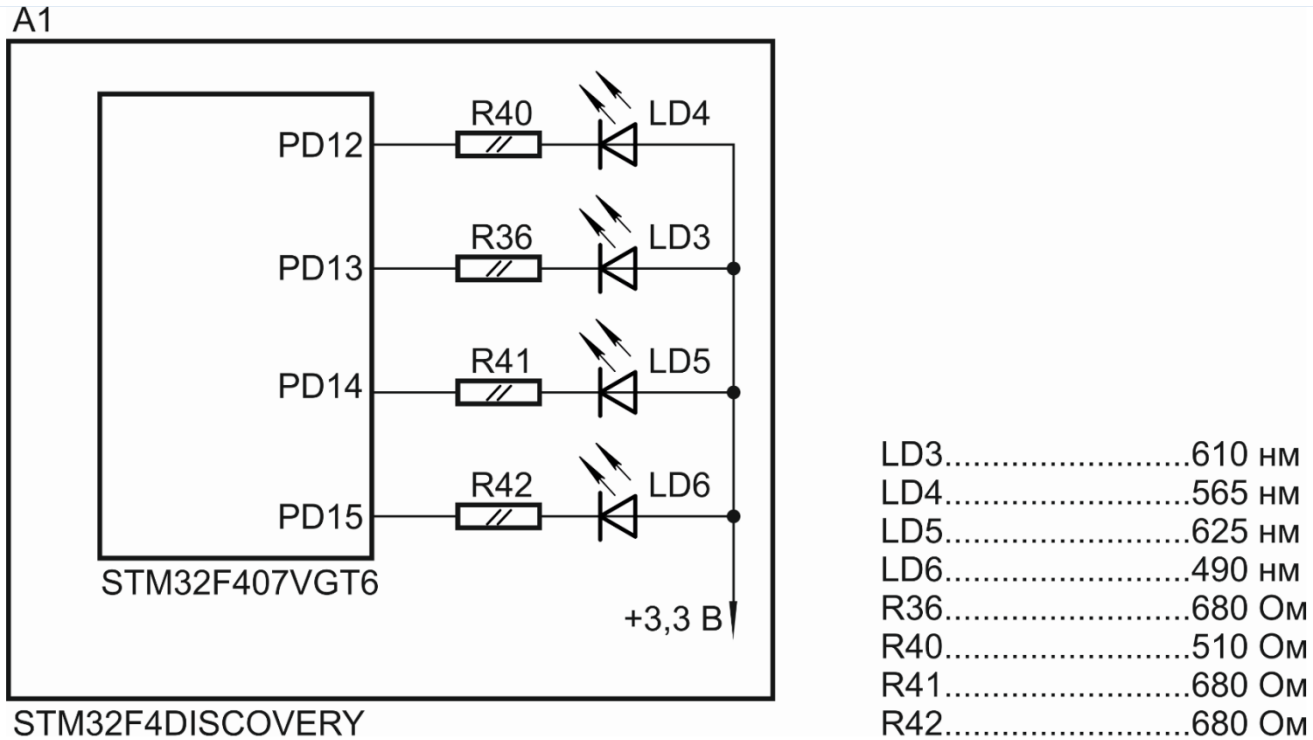


Рисунок 15 – Схема подключения светодиодов к контроллеру

Текст программы.

```
#include "stm32f4xx.h"

void Delay(long i); //Описание прототипа функции подпрограммы задержки

int main(void)
{

    RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN; // Задание тактовой частоты порта D

    GPIOD->MODER = 0x55000000; // Контакты с PD12 по PD15 сконфигурированы как
//выходы

    while(1)
    {
        GPIOD->BSRRH = 0xF000; // все светодиоды выключены
        Delay(1000000);                    //вызов подпрограммы задержки
        GPIOD->BSRRL = 0x1000; // включен светодиод LD4
        Delay(1000000);                   //вызов подпрограммы задержки
        GPIOD->BSRRL = 0x2000; // включен светодиод LD3
        Delay(1000000);                   //вызов подпрограммы задержки
        GPIOD->BSRRL = 0x4000; // включен светодиод LD5
        Delay(1000000);                   //вызов подпрограммы задержки
        GPIOD->BSRRL = 0x8000; // включен светодиод LD6
        Delay(1000000);                   //вызов подпрограммы задержки
    }
}

void Delay(long I) //Подпрограмма задержки
{
    volatile long iZ = i; // Объявление рабочей переменной
    while (iZ--);
}
```

Как видно из текста, программа выполняет следующее. Вначале происходит подключение библиотеки, в которой содержатся все необходимые переменные и функции для работы с контроллером. Далее идёт описание прототипа функции, которая используется для задержки выполнения программы на определённое время. После идет основной цикл программы, в начале которого задается тактовая частота порта D. Она задается при помощи ранее определенный переменной, которая имеет такое значение, что в результате выполнения операции «исключающее или» в поле AHB1ENR структуры RCC происходит запись значения, которое определяет подачу тактовой частоты на порт D. Далее происходит конфигурирование контактов порта D, контакты которого с 12 по 15 в результате проведённой операции конфигурируются как выходы. Затем начинается цикл, в котором постоянно выполняется следующее.

Вначале все контакты порта D, связанные со светодиодами, переводятся в состояние логической единицы. Это приводит к тому, что светодиоды LD3, LD4, LD5 и LD6 гаснут. Далее вызывается подпрограмма задержки, а затем контакт контроллера, связанный со светодиодом LD4, переводится в состояние низкого уровня — это приводит к тому, что возникает разность потенциалов, и светодиод начинает светиться. После этого вызывается подпрограмма задержки и аналогичным образом зажигается светодиод LD3, и аналогично — LD5 и LD6, после чего цикл повторяется.

Сама подпрограмма задержки представляет собой программу, которая занимает время контроллера. Фактически она производит обратный отчет от заданного числа до нуля. Это продолжается некоторое время, которое и является тем временем, на которое объявляется задержка.

Для того, чтобы загрузить программу в память контроллера, необходимо вначале при помощи кабеля из комплекта поставки отладочной платы соединить её с ПЭВМ согласно схеме, представленной на рисунке 16.

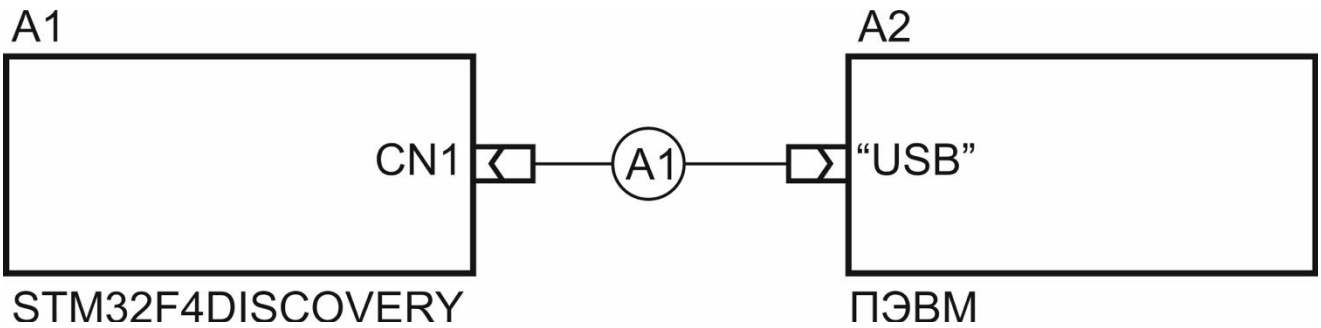


Рисунок 16 – Схема соединения отладочной платы с ПЭВМ

Далее необходимо выполнить собственно загрузку программы в ЭСППЗУ контроллера. Вначале в главном меню следует выбрать **View → Configuration** и в открывшемся окнее в меню **Debugger** выбрать адаптер **ST-LINK**. Далее в главном меню необходимо выбрать **Flash → Program Download**

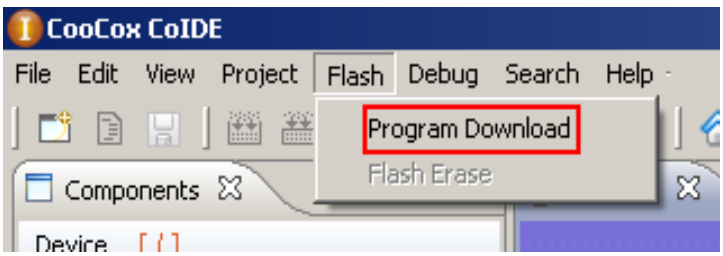


Рисунок 17 – (пояснение в тексте)

(или на панели инструментов нажать иконку «Download Code To Flash», рисунок 18). Программа будет загружена в контроллер. Затем следует нажать кнопку **F5**.

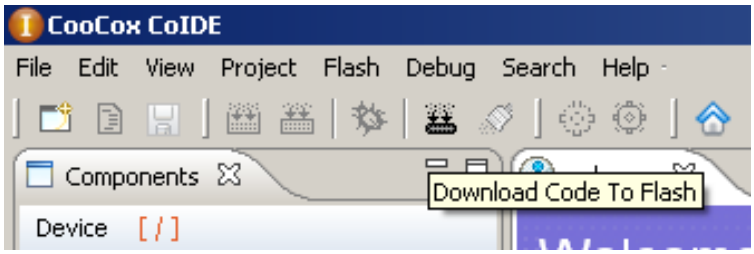


Рисунок 18 – (пояснение в тексте)

В результате на плате наблюдается поочерёдное включение светодиодов LD4, LD3, LD5 и LD6 с их последующим затуханием (рисунок 19).

Рисунок 19 – (пояснение в тексте)

Также в CoIDE доступен режим отладки. Для доступа к нему необходимо нажать **CTRL+F5** (или выбрать в главном меню **Debug → Debug**).

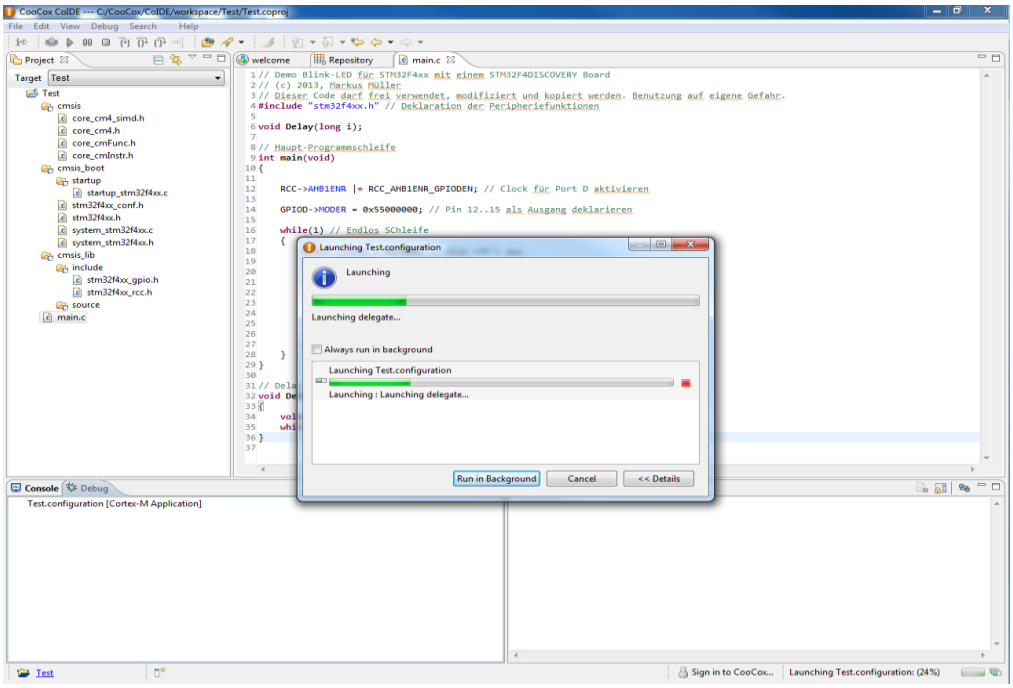


Рисунок 20 – Загрузка и запуск программы на отладочной плате

Контрольные вопросы:

1. Что такое STM32F4Discovery?
2. На базе какого контроллера разработана STM32F4Discovery?
3. Для чего нужна среда CooCox CoIDE?
4. Для чего нужен компилятор?

2 Лабораторная работа №2. Написание тестового проекта.

2.2 Введение

В ходе лабораторной работы будет создана программа, которая будет выполнять простейшую задачу, а именно — задавать прерывистое свечение светодиода. При этом отладочная плата будет доработа: к контакту PD1 будет подключен светодиод.

2.3 Описание лабораторной работы

Цели работы

1. Ознакомиться с приёмами работы при программировании ARM-контроллера;
2. Ознакомиться с приёмами отладки и контроля цифровых схем

Указания к выполнению работы

Лабораторная работа состоит из двух частей. Первая часть посвящена сборке электрической схемы. Вторая часть подразумевает собственно программирование ARM-контроллера и работу со средствами контроля.

Порядок выполнения работы

Часть 1:

1. Выясните местоположение контактов согласно КД на отладочную плату STM32F4 Discovery;
2. Соберите схему.

Часть 2:

1. Запрограммируйте контроллер;
2. При помощи осциллографа проконтролируйте наличие и форму сигнала, получаемого с контроллера;

Требования к отчету

Отчёт должен быть оформлен в соответствии с требованиями содержащимся в приложении А.

Отчет должен включать:

1. Титульный лист
2. Цели работы
3. Отчёт о выполнении задания части 1:
   - — Электрическая схема;
   - — Схема подключения лабораторного оборудования.
4. Отчёт о выполнении заданий части 2:
   - — Код программы;
   - — UML диаграмма действий, описывающая алгоритм работы программы;
   - — Показания осциллографа.
5. Выводы по работе.

2.4 Выполнение работы

Для данного примера разница между предыдущей схемой и STM32F4DISCOVERY фактически только в номере контакта, к которому подключен светодиод, то есть используется первый контакт порта GPIOD.

Из документации на отладочную плату можно выяснить, как следует подключать светодиод.

Информация о подключениях представлена в следующих местах:

- в параграфе 4.4, стр. 16 [11];
- в таблице 5, на стр. 30 [11];
- на рисунке 16, стр. 40 [11].

Светодиод подключается к отладочной плате так, как показано на рисунке 21.

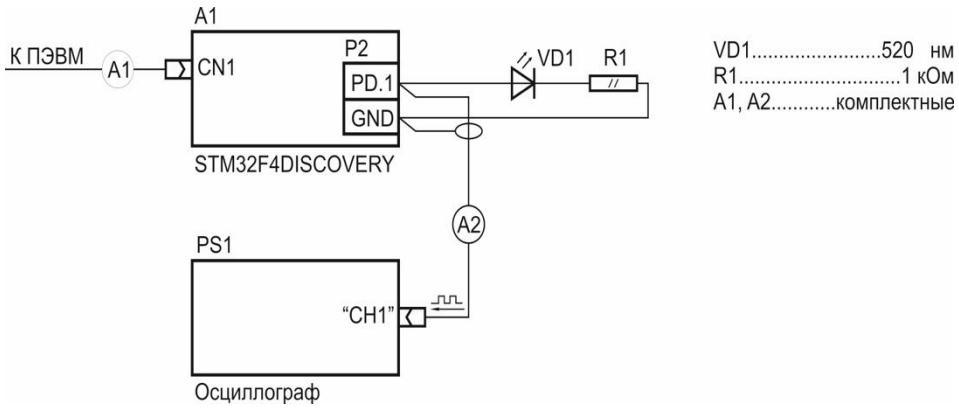


Рисунок 21 – Схема подключений лабораторного оборудования

Для того, чтобы задать прерывистое свечение светодиода, необходимо воспроизвести следующий алгоритм.

1. Подать на выход PD1 высокий логический уровень,
2. Выдержать паузу,
3. Подать на выход PD1 низкий логический уровень,
4. Выдержать паузу,
5. Перейти к началу.

Функция задержки основана на пустом цикле и аналогична таковой из первой лабораторной работы. Основное её предназначение — занятие времени контроллера.

В новой программе используется порт GPIOD, поэтому строчка с тактированием порта остаётся нетронутой. Затем в поле **MODER** структуры порта **GPIOD** записывается значение **0x4**, это конфигурирует первый контакт порта D как выход. Отметим, что каждое из состояний контакта порта может быть представлено в двоичном виде. Таких состояний несколько — высокоимпедансное, вход, выход и работа в режиме альтернативного функционирования. Очевидно, что для их кодирования необходимо два бита. В нашем случае состояние первого контакта порта представлено комбинацией 01.

Далее происходит обращение к полю **BSRRH** структуры **GPIOD** и запись в него значения **0x2**. Это приводит к тому, что на контакте порта выставляется высокий уровень напряжения, равный 3,3 В. Затем происходит вызов подпрограммы задержки. Далее происходит обращение к полю **BSRRL** структуры **GPIOD** и запись в него значения **0x2**. Это приводит к тому, что на контакте порта выставляется низкий уровень напряжения. Остальная часть программы остается без изменений.

Текст программы:

```
#include "stm32f4xx.h"

void Delay(long i);

int main(void)
{

    RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN; // Включено тактирование
                                         //порта D
    GPIOD->MODER = 0x4; // Первый контакт сконфигурирован на выход

    while(1)
    {
        GPIOD->BSRRH = 0x2; // Подача высокого уровня на PD1
        Delay(1000000);             // Пауза
        GPIOD->BSRRL = 0x2; // Подача низкого уровня на PD1
        Delay(1000000);             // Пауза
    }
}

void Delay(long i)
{
    volatile long iZ = i;
    while (iZ--);
}
```

Для более наглядной демонстрации работы платы можно изменить задержку между включением и выключением светодиода. Осцилограмма сигнала снимается с осциллографа PS1 (рисунок 22).

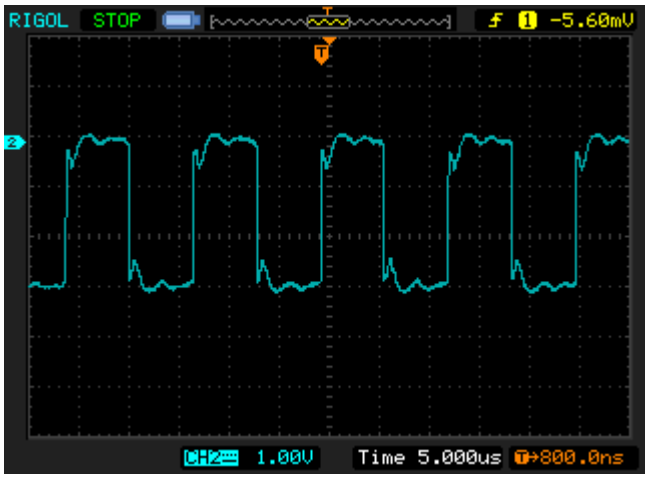


Рисунок 22 – Осциллограмма сигнала

Искажение сигнала объясняется подключенными в цепь резистором и диодом.

Контрольные вопросы:

1. Сколькими битами описывается состояние контакта порта?
2. Для чего служат порты контроллера stm32f407VGT6?
3. Как организуется задержка в тестовой программе?
4. Какой протокол используется для передачи программы в контроллер?

3 Лабораторная работа № 3. Таймеры и прерывания

3.2 Введение

В ходе данной работы будет изучена возможность создания временной задержки с использованием прерывания по таймеру.

Прерывание – это программный механизм, который позволяет аппаратному обеспечению сообщать о наступлении определенных событий в своей работе. В момент, когда происходит прерывание, процессор переключается с выполнения основной программы на выполнение соответствующего обработчика прерываний. Как только выполнение обработчика завершено, продолжается выполнение основной программы с места, в котором она была прервана.

В случае использования прерывания программа не требует постоянного контроля работы со стороны внешнего источника, поскольку контроль самого события, приводящего к прерыванию, происходит средствами контроллера. При этом программист избавляется от необходимости постоянно отслеживать такое

событие.

Для работы с прерываниями контроллер STM32F407VGT6 имеет встроенный контроллер векторов прерываний (Nested vectored interrupt controller, NVIC).

Встроенный контроллер векторов прерываний управляет 16 уровнями приоритетов и может обрабатывать до 82 каналов маскируемых прерываний, а также 16 источников прерываний контроллера Cortex M4. Особенности встроенного контроллера векторов прерываний заключаются в следующем:

- Возможность быстрой обработки прерываний.
- Каждый вектор прерываний в таблице прерываний имеет доступ прямо к ядру.
- Возможность обработки прерываний с высоким приоритетом, которые произошли позднее, чем обрабатываемое.
- Ранняя обработка прерываний.
- Автоматическое сохранение состояния процессора.
- Восстановление состояния до прерывания по выходу из обработчика без дополнительных инструкций. Данный блок аппаратного обеспечения обеспечивает гибкое управление прерываниями с минимальными задержками на обработку прерываний.

Контроллер внешних прерываний (External interrupt/event controller (EXTI))

Контроллер внешних прерываний состоит из 23 линий, которые фиксируют переход по фронту и используются для запрашивания прерываний либо обработки событий. Каждая линия может быть независимо сконфигурирована для обработки определённого вида сигнала по фронту, по спаду или по обоим из них, а также может быть независимо маскирована. Состоянием обработчика прерываний управляет специальный регистр обработки. Контроллер внешних прерываний может обнаруживать событие на внешней линии за время меньшее, чем один такт.

С каждой из 16 линий прерываний может быть связан любой из 82 контактов портов контроллера.

Контроллер STM32F407VGT6 имеет несколько таймеров, которые могут быть задействованы пользователем. К их числу относятся два таймера с дополнительным управлением, восемь таймеров общего назначения, два базовых таймера и два сторожевых таймера. В режиме отладки каждый из таймеров может быть переведен в останов. Рассмотрим различные типы таймеров подробнее.

Таймеры с дополнительным управлением (TIM1, TIM8)

Таймеры с дополнительным управлением (TIM1, TIM8) могут рассматриваться как трёхфазные генераторы ШИМ с шестиканальным

мультиплексированием. У них есть дополнительные выходы ШИМ с программируемыми задержками. Также они могут рассматриваться в качестве таймеров общего назначения. У каждого таймера имеется 4 независимых канала, которые могут быть использованы для:

— захвата по входу
— сравнения по выходу
— генерирования сигнала ШИМ с выравниваем по фронту, либо по середине импульса
— генераторов одиночного сигнала.

Если эти таймеры сконфигурированы как обычные 16-битные таймеры, то они обладают теми же самыми особенностями, что и таймеры общего назначения. Если их сконфигурировать как 16-битные генераторы ШИМ, то в этом случае они обладают полным диапазоном глубины модуляции от нуля до 100%. Таймеры с дополнительным управлением могут работать совместно с таймерами общего назначения с использованием опции Timer Link для синхронизации или отслеживания событий. Таймеры TIM1 и TIM8 поддерживают независимый доступ к памяти.

Таймеры общего назначения

У контроллера STM32F407VGT6 имеется 10 синхронизируемых таймеров общего назначения. Рассмотрим особенности их использования.

Контроллер STM32F407VGT6 включает в себя 4 полностью управляемых таймера: TIM2, TIM5, TIM3 и TIM4. Таймеры TIM2 и TIM5 основаны на 32-битных перезагружаемых счётчиках с возможностью прямого и обратного отсчета и 16-битных делителях. Таймеры TIM3 и TIM4 основаны на 16-разрядных перезагружаемых счетчиках с возможностью прямого и обратного отсчета и 16-битных делителях. У каждого из таймеров имеется 4 независимых канала для захвата по входу или сравнения по выходу, ШИМ или работы в режиме одиночного импульса. Это позволяет построить на их основе до 16 схем захвата по входу или сравнения по выходу, или ШИМ. Таймеры TIM2, TIM3, TIM4 и TIM5 представляют собой таймеры общего назначения, которые могут работать совместно с другими таймерами общего назначения или с таймерами TIM1 и TIM8 посредством использования опции Timer Link для синхронизации или отслеживания событий.

Каждый из этих таймеров общего назначения может использоваться для генерации сигналов ШИМ. Таймеры TIM2, TIM3, TIM4 и TIM5 имеют независимый механизм формирования запросов памяти. Также с их помощью возможна обработка сигналов, поступающих с датчиков угла поворота или цифровых данных, которые поступают с одного, либо двух, либо трёх, либо четырёх датчиков Холла.

Таймеры TIM9, TIM10, TIM11, TIM12, TIM13 и TIM14

Таймеры TIM9, TIM10, TIM11, TIM12, TIM13 и TIM14 основаны на 16-битных счётчиках с возможностью прямого и обратного отсчета и 16-битных делителях. Каждый из таймеров TIM10, TIM11, TIM13 и TIM14 имеет по одному независимому каналу, в то время как таймеры TIM9 и TIM12 имеют два независимых канала для захвата по входу либо сравнения по выходу, работы в режиме ШИМ или в режиме одиночного импульса. Также они могут быть синхронизированы с таймерами TIM2, TIM3, TIM4 и TIM5 как таймеры общего назначения. Помимо этого, они могут использоваться в качестве источников сигнала времени.

Таймеры TIM6 и TIM7

Эти таймеры в основном используются для работы с цифро-аналоговым преобразователем и формирования сигналов специальной формы. Они могут быть использованы как 16-битные источники сигнала времени, а также поддерживают механизм независимого запроса памяти.

3.3 Описание лабораторной работы

Цели работы

1. Ознакомиться с понятием прерывания и возможностями, которые предоставляют прерывания;
2. Научиться использовать таймеры для организации задержки по времени.

Указания к выполнению работы

Лабораторная работа состоит из двух частей. В ходе выполнения первой части производится настройка проекта и программирование ARM – контроллера. В ходе выполнения второй части производится контроль выходного сигнала и отладки программы.

Порядок выполнения работы

Часть 1:

1. Настройте проект;
2. Напишите программу.

Часть 2:

1. Соберите схему;
2. Загрузите программу;
3. Проконтролируйте выходной сигнал при помощи осциллографа.

Требования к отчету

Отчет должен быть оформлен в соответствии с требованиями, представленными в приложении А.

Отчет должен включать:

1. Титульный лист;
2. Цели работы;
3. Отчёт о выполнении заданий части 1:
   — Электрическая схема;
   — Схема подключения лабораторного оборудования;
4. Отчёт о выполнении заданий части 2:
   — Код программы;
   — UML диаграмма действий, описывающая алгоритм работы программы;
   — Показания дисплея;
5. Выводы по работе.

3.4 Выполнение работы

Для того, чтобы задействовать таймер, его вначале необходимо сконфигурировать. Для этого необходимо выполнить следующее:

— включить тактирование таймера;

— задать делитель и период счёта;

Для того, чтобы задействовать прерывание по таймеру, его необходимо включить, а также назначить данному прерыванию свой обработчик. Для этого необходимо выполнить следующее.

— Разрешить прерывание по определенному событию таймера: переполнению, переходу в нуль, либо достижению границы счёта.

— Разрешить обработку прерывания по указанному событию.

Схема подключений лабораторного оборудования приведена на рисунке 23.

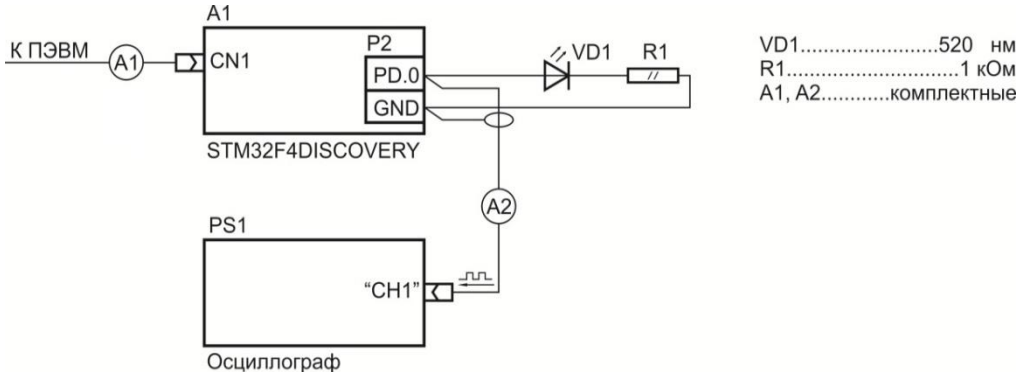


Рисунок 23 – Схема подключений лабораторного оборудования

Для того, чтобы задать прерывистое свечение светодиода, необходимо воспроизвести следующий алгоритм:

1. Подать на выход PD1 высокий логический уровень,
2. Ждать, пока не произойдёт прерывание по таймеру,
3. Подать на выход PD1 низкий логический уровень,
4. Ждать, пока не произойдёт прерывание по таймеру,
5. Перейти к пункту №1.

Исходный код программы

```
#include "stm32f4xx.h"
#include <stm32f4xx_rcc.h>
#include <stm32f4xx_gpio.h>
#include "C:\CooCox\CoIDE\cmsis_lib\Include\stm32f4xx_tim.h"
#include <misc.h>

int a = 0;

// Основная программа
int main(void)
{

    RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN; // Подача тактовой частоты на
//порт D

    GPIOD->MODER = 0x1; // Нулевой контакт порта D сконфигурирован как
//выход

    init_timer();

    while(1)
    {
     while (!a)
     {GPIOD->BSRRH = 0x1; // LED pd0 off
     }
        GPIOD->BSRRL = 0x1; // LED pd0 on

    }
}

// Подпрограмма задержки

void init_timer(){
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE); // включаем
//тактирование таймера
```

```
   /* Другие параметры структуры TIM_TimeBaseInitTypeDef
    * не имеют смысла для базовых таймеров.
    */

     TIM_TimeBaseInitTypeDef base_timer;
     TIM_TimeBaseStructInit(&base_timer);
  /* Делитель учитывается как TIM_Prescaler + 1, поэтому вычитаем 1 */
     base_timer.TIM_Prescaler = 24000 - 1; // делитель 24000
     base_timer.TIM_Period = 10; //период 10 импульсов
     TIM_TimeBaseInit(TIM6, &base_timer);

  /* Разрешаем прерывание по обновлению (в данном случае -
   * по переполнению) счётчика таймера TIM6.
   */

     TIM_ITConfig(TIM6, TIM_IT_Update, ENABLE);
     TIM_Cmd(TIM6, ENABLE);  // Включаем таймер

  /* Разрешаем обработку прерывания по переполнению счётчика
   * таймера TIM6. это же прерывание
   * отвечает и за опустошение ЦАП.
   */

     NVIC_EnableIRQ(TIM6_DAC_IRQn);
}

void TIM6_DAC_IRQHandler(){

  /* Так как этот обработчик вызывается и для ЦАП, нужно проверять,
   * произошло ли прерывание по переполнению счётчика таймера TIM6.
   */
     if (TIM_GetITStatus(TIM6, TIM_IT_Update) != RESET) {
     /* Очищаем бит обрабатываемого прерывания */
       TIM_ClearITPendingBit(TIM6, TIM_IT_Update);

       a = ~a;
     }
}
```

Отметим, что для корректной сборки программы необходимо добавить в директорию проекта файлы «stm32f4xx_tim.h» и «stm32f4xx_tim.c», предварительно загруженные из Интернет. Затем файл «stm32f4xx_tim.h» копируется в папку cmsis (рисунок 24).

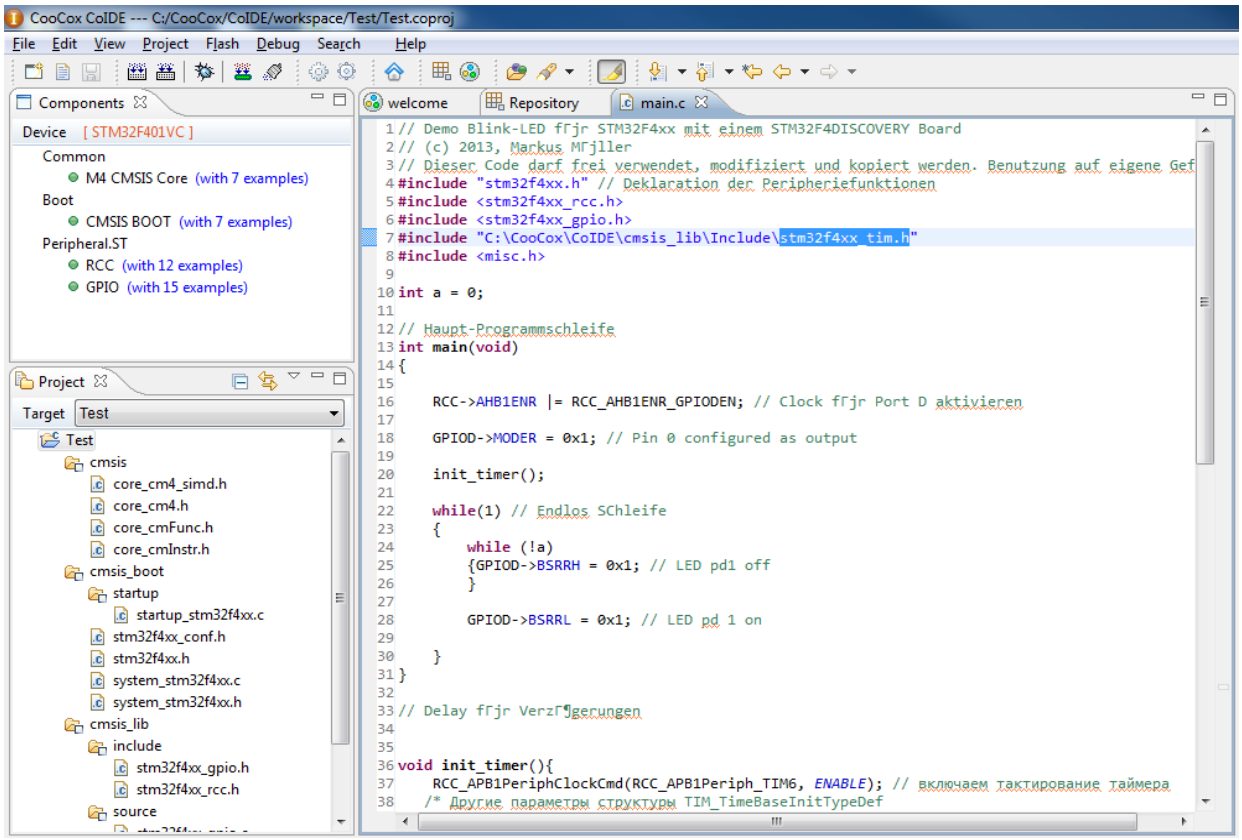


Рисунок 24 – (пояснение в тексте)

Для того, чтобы добавить файл, нажатием правой кнопки мыши на директории проекта следует выбрать пункт «Add Files», затем в появившемся окне выбрать файл «stm32f4xx_tim.h» (рисунок 25).

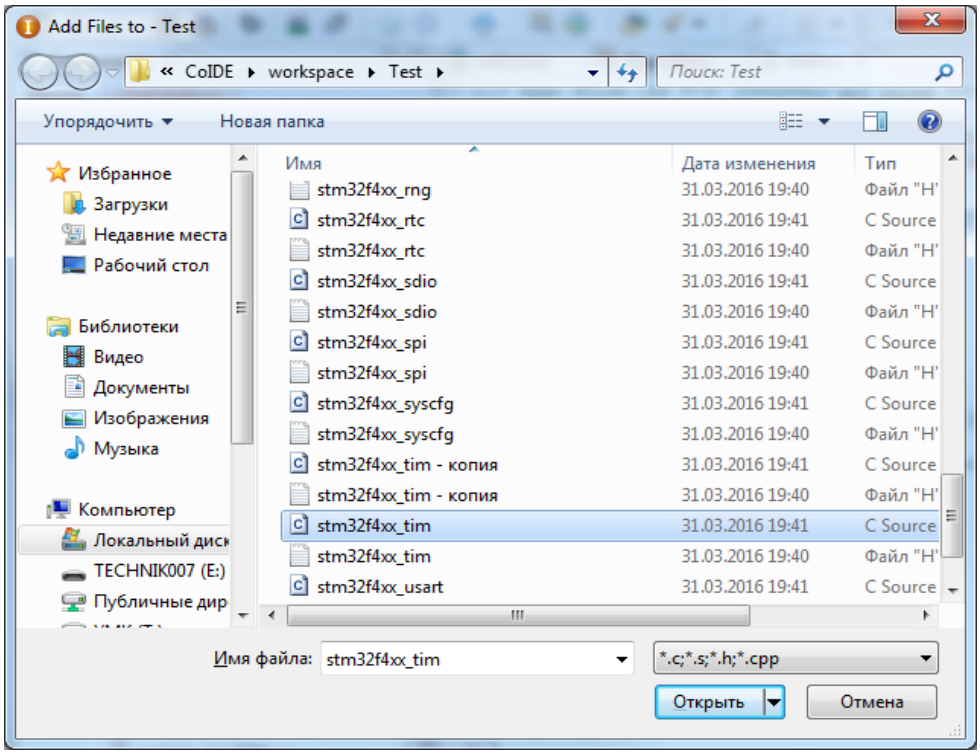


Рисунок 25 – Добавление файла в проект

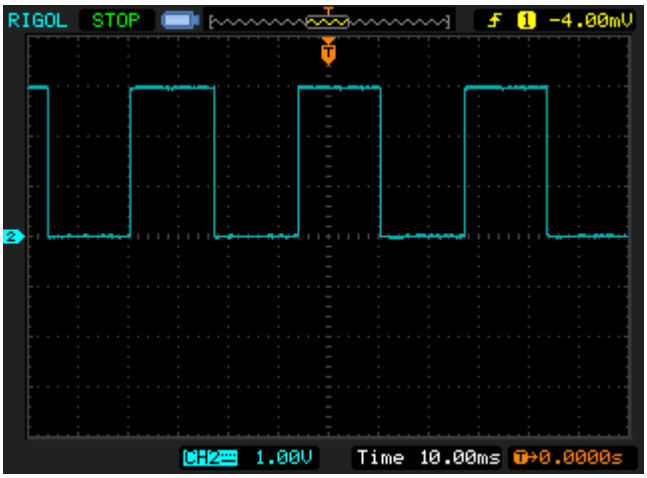


Рисунок 26 – Осциллограмма сигнала

Вид поступающего на осциллограф сигнала приведен на рисунке 26. Отсутствие искажений сигнала обусловлено снятием осциллограммы при неподключенных резисторе и диоде.

Контрольные вопросы

1. Что такое прерывание?
2. Что такое таймер?

3. Сколько всего таймеров имеет контроллер STM32F407VGT6? Назовите типы этих таймеров.
4. Как конфигурируется таймер?

4 Лабораторная работа №4. Вывод данных на семисегментный индикатор

4.2 Введение

Отображение информации является важной частью любого информационного процесса измерения, управления и контроля. Воспринимаемая человеком информация должна быть представлена в удобной форме с минимальными требованиями к перекодировке. Поэтому широкое распространение получило знаковое представление информации.

В цифровой технике применяют индикаторы, основанные на различных принципах действия. Наиболее распространённым и удобным средством отображения информации являются семисегментные индикаторы (ССИ). Семь отображающих элементов позволяют высвечивать десятичные и шестнадцатеричные цифры, ряд букв русского и латинского алфавитов, а также некоторые специальные знаки.

Для отображения многосимвольной информации используются линейные дисплеи. Такой дисплей представляет собой линейку, смонтированную из отдельных ССИ. Число знакомест дисплея определяется в соответствии с требованиями к проектируемой системе.

4.3 Описание лабораторной работы

Цели работы

1. Освоить вывод данных с контроллера на семисегментный индикатор;
2. Освоить использование дополнительного оборудования, подключаемого к контроллеру.

Указания к выполнению работы

Лабораторная работа состоит из двух частей. В ходе выполнения первой части производится сборка электрической схемы. При выполнении второй части производится программирование контроллера и отладка программы.

Порядок выполнения работы

Часть 1:

— Выясните местоположение необходимых контрактов согласно КД на отладочную плату STM32F4 Discovery;

— Соберите схему.

Часть 2:

— Запрограммируйте контроллер;

— Проконтролируйте выполнение программы.

Требования к отчету

Отчет должен быть оформлен в соответствии с требованиями, представленными в Приложении А.

Отчет должен включать:

1. Титульный лист
2. Цели работы
3. Отчет о выполнении заданий части 1:
   — Электрическая схема;
   — Схема подключения лабораторного оборудования.
4. Отчёт о выполнении заданий части 2:
   — Код программы;
   — UML – диаграмма действий, описывающая алгоритм работы программы;
   — Показания дисплея.
5. Выводы по работе.

4.4 Выполнение работы

Структуру ССИ можно показать на примере индикатора типа АЛС321А аА0.336.245ТУ (рисунок 27).

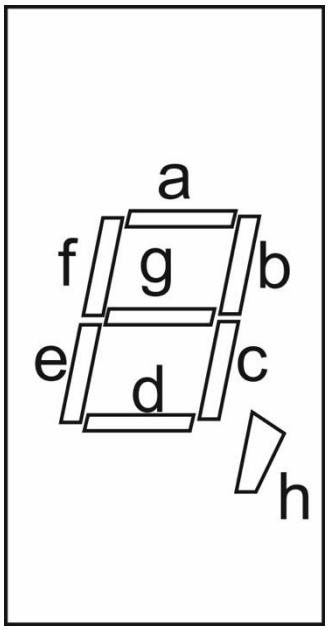


Рисунок 27 – Расположение знакосинтезирующих элементов на индикаторе АЛС321А

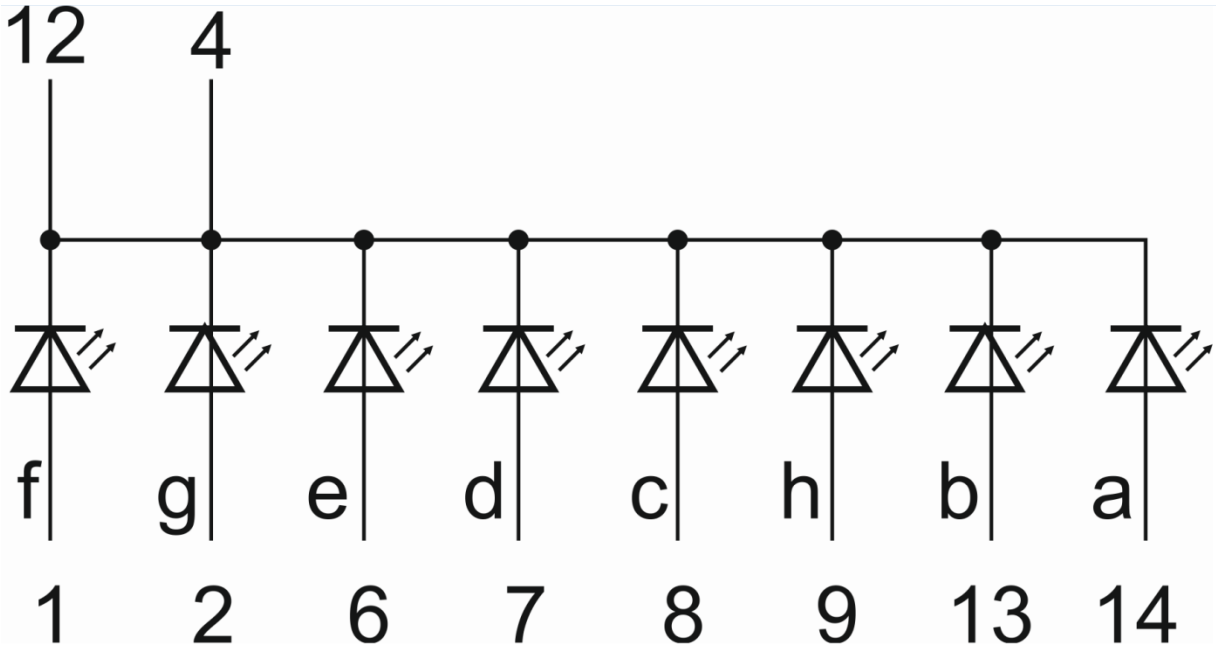


Рисунок 28 – Электрическая схема индикатора АЛС321А

Индикатор имеет семь элементов и десятичную точку, излучающих свет при подаче прямого тока. Различные комбинации элементов, обеспечиваемые внешней коммутацией, позволяют воспроизвести цифры от 0 до 9 с точкой или без неё. Все элементы индикатора АЛС321А имеют общий катод. Параметры индикатора представлены в таблицах 1 и 2.

Таблица 1. Параметры индикатора АЛС321А

Цвет свечения	Желто-зеленый
Сила света при постоянном токе 20 мА через элемент, не менее: элемента	 12 мкд
децимальной точки	0,02 мкд
Относительная неравномерность силы света между элементами, не более	3
Постоянное прямое напряжение при Iпр = 20 мА, не более:	
Т = +25 и +70 °С	3,6 В
Т = -60 °С	4 В

Таблица 2. Предельные эксплуатационные данные индикатора АЛС321А

Постоянное (импульсное) обратное напряжение излучающего элемента	5 В
Постоянный прямой ток через элемент: Т = -60...+35 °С	 25 мА
Т = +70 °С	17,5 мА
Рассеиваемая мощность: Т = -60...+35 °С	720 мВт
Т = +70 °С	400 мВт
Температура окружающей среды	-60...+70 °С

Для работы с семисегментным индикатором возможно использование двух различных схем. Первая из них подразумевает управление каждым из сегментов индикатора отдельно. При этом сегменты подключаются к определенным контактам порта контроллера. В этом случае необходимо использовать подпрограмму, которая будет переводить текущее значение в определённую комбинацию сегментов и выводить эту комбинацию в порт. Вторая схема подразумевает использование специальной микросхемы — дешифратора.

Для кодирования и декодирования информации в цифровой электронике используются шифраторы и дешифраторы. Шифратор — цифровое устройство, которое преобразует сигналы, поданные на один из его входов, в кодовые

комбинации на выходах (параллельный двоичный код), каждая из которых соответствует только одному выбранному входу.

Дешифратор — цифровое устройство, которое выполняет операцию, обратную шифрации, т.е. переводит двоичный код, поданный на его входы, в сигнал только на том выходе, который соответствует поданному коду. В качестве дешифраторов могут выступать специализированные микросхемы, а также другие элементы цифровой электроники. Так, дешифратором может служить программируемое постоянное запоминающее устройство (ППЗУ).

В ходе данной лабораторной работы используется дешифратор — микросхема HEF4511BC (рисунок 29), к которой подключается индикатор АЛС321А.

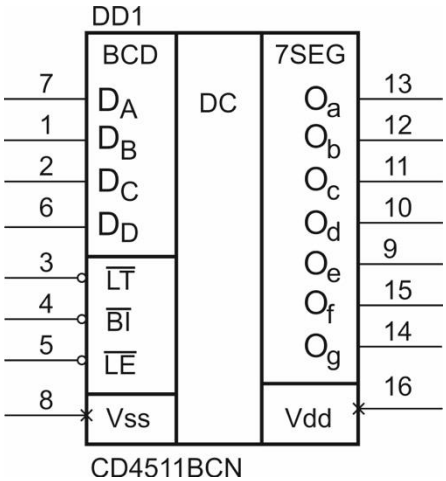
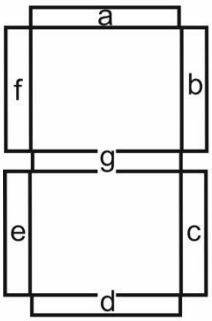


Рисунок 29 – Назначение контактов микросхемы HEF4511BC

Представленный на рисунке 29 дешифратор переводит четырёхбитное двоичное число в диапазоне 0...9 в десятичную цифру. Схема расположения сегментов индикатора и таблица соответствия четырёхбитных кодов логическим уровням на выходах дешифратора приведены на рисунке 30.



$D_D D_C D_B D_A$	a	b	c	d	e	f	g
0 0 0 0	1	1	1	1	1	1	0
0 0 0 1	0	1	1	0	0	0	0
0 0 1 0	1	1	0	1	1	0	1
0 0 1 1	1	1	1	1	0	0	1
0 1 0 0	0	1	1	0	0	1	1
0 1 0 1	1	0	1	1	0	1	1
0 1 1 0	0	0	1	1	1	1	1
0 1 1 1	1	1	1	0	0	0	0
1 0 0 0	1	1	1	1	1	1	1
1 0 0 1	1	1	1	0	0	1	1

Рисунок 30 – Схема расположения сегментов индикатора

Рассмотрим назначение контактов дешифратора подробнее.

D_A...D_D — входы данных

$\overline{LE}$ — вход разрешения захвата (для включения следует подать 0 В)

$\overline{BI}$ — вход разрешения мерцания (для включения следует подать 0 В)

$\overline{LT}$ — вход проверки индикатора (для включения следует подать 0 В)

O_a...O_g — выходы на сегменты индикатора.

Таблица 3. Таблица истинности дешифратора HEF4511BC

Входы							Выходы							Отображаемое значение
$\overline{LE}$	$\overline{BI}$	$\overline{LT}$	D_D	D_C	D_B	D_A	O_a	O_b	O_c	O_d	O_e	O_f	O_g	
X	X	0	X	X	X	X	1	1	1	1	1	1	1	8
X	0	1	X	X	X	X	0	0	0	0	0	0	0	мерцание
0	1	1	0	0	0	0	1	1	1	1	1	1	0	0
0	1	1	0	0	0	1	0	1	1	0	0	0	0	1
0	1	1	0	0	1	0	1	1	0	1	1	0	1	2
0	1	1	0	0	1	1	1	1	1	1	0	0	1	3
0	1	1	0	1	0	0	0	1	1	0	0	1	1	4
0	1	1	0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	1	0	0	0	1	1	1	1	1	6
0	1	1	0	1	1	1	1	1	1	0	0	0	0	7
0	1	1	1	0	0	0	1	1	1	1	1	1	1	8
0	1	1	1	0	0	1	1	1	1	0	0	1	1	9
0	1	1	1	0	1	0	0	0	0	0	0	0	0	мерцание

0	1	1	1	0	1	1	0	0	0	0	0	0	0	мерцание
0	1	1	1	1	0	0	0	0	0	0	0	0	0	мерцание
0	1	1	1	1	0	1	0	0	0	0	0	0	0	мерцание
0	1	1	1	1	1	0	0	0	0	0	0	0	0	мерцание
0	1	1	1	1	1	1	0	0	0	0	0	0	0	мерцание
1	1	1	X	X	X	X	*							*

Примечания:

1) X — состояние не имеет значения

2) * — значение зависит от числа, поданного на вход в момент перехода сигнала на контакте $\overline{LE}$ из низкого уровня в высокий

Для работы с индикатором будет использоваться только один разряд. В этом случае программного преобразования данных не требуется, поскольку мы задействуем только младшую тетраду. Схема подключения индикатора к плате STM32F4 Discovery приведена на рисунке 31.

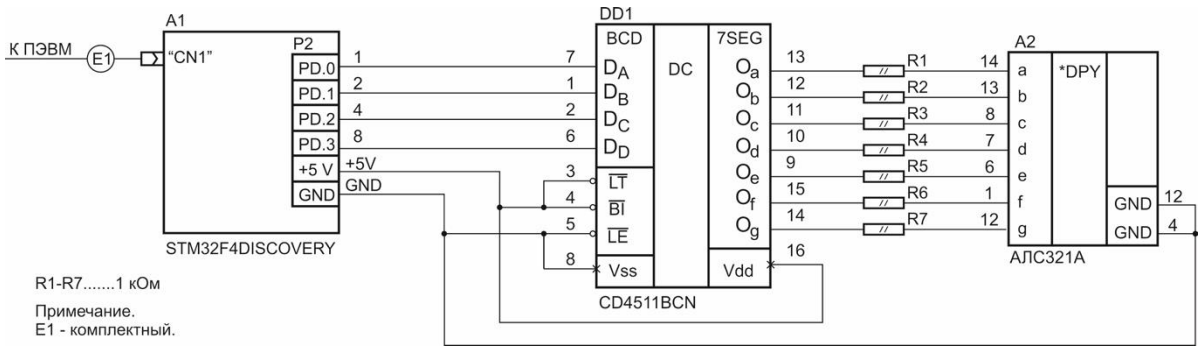


Рисунок 31 – Схема подключения оборудования к плате STM32F4 Discovery

Для того, чтобы продемонстрировать работу схемы, в качестве примера предлагается программа, реализующая отсчёт от нуля до девяти. Для того, чтобы задать интервал отсчёта, будет использоваться прерывание по таймеру, рассмотренное в предыдущей лабораторной работе.

Для того, чтобы реализовать отсчёт от нуля до девяти, необходимо воспроизвести следующий алгоритм:

1. Задать переменную с нулевой величиной.
2. Подать эту переменную на выход порта D,
3. Выдержать паузу,
4. Прибавить к переменной единицу,
5. Проверить, не вышло ли значение переменной за границы диапазона, и если вышло, то обнулить его,
6. Перейти к пункту № 2.

Текст программы:

```
#include "stm32f4xx.h"
#include <stm32f4xx_rcc.h>
#include <stm32f4xx_gpio.h>
#include "C:\CooCox\CoIDE\cmsis_lib\Include\stm32f4xx_tim.h"
#include <misc.h>

int a = 0;

int main(void)
{

    RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN; // Подача тактовой частоты на
//порт D

    GPIOD->MODER = 0x55; // Контакты PD.0...PD.3 сконфигурированы как
//выходы

    init_timer();

    while(1) // главный цикл
    {
        if(a == 9) a=0;

        GPIOD->ODR = a;

    }
}

// Работа со временными задержками

void init_timer(){
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE); // включаем
//тактирование таймера
   /* Другие параметры структуры TIM_TimeBaseInitTypeDef
   * не имеют смысла для базовых таймеров.
   */
```

```
    TIM_TimeBaseInitTypeDef base_timer;
    TIM_TimeBaseStructInit(&base_timer);
  /* Делитель учитывается как TIM_Prescaler + 1, поэтому отнимаем 1 */
    base_timer.TIM_Prescaler = 24000 - 1; // делитель 24000
    base_timer.TIM_Period = 1500; //период 1500 импульсов
    TIM_TimeBaseInit(TIM6, &base_timer);

  /* Разрешаем прерывание по обновлению (в данном случае -
   * по переполнению) счётчика таймера TIM6.
   */

    TIM_ITConfig(TIM6, TIM_IT_Update, ENABLE);
    TIM_Cmd(TIM6, ENABLE);  // Включаем таймер

  /* Разрешаем обработку прерывания по переполнению счётчика
   * таймера TIM6. это же прерывание
   * отвечает и за опустошение ЦАП.
   */

    NVIC_EnableIRQ(TIM6_DAC_IRQn);
}

void TIM6_DAC_IRQHandler(){

  /* Так как этот обработчик вызывается и для ЦАП, нужно проверять,
   * произошло ли прерывание по переполнению счётчика таймера TIM6.
   */
    if (TIM_GetITStatus(TIM6, TIM_IT_Update) != RESET) {
    /* Очищаем бит обрабатываемого прерывания */
      TIM_ClearITPendingBit(TIM6, TIM_IT_Update);

      a++;
    }
}
```

Контрольные вопросы

1. Что такое дешифратор?
2. В каком случае может потребоваться перекодировка числового значения перед его индикацией?
3. Как можно обойтись в рассмотренном проекте без использования дешифратора?
4. Как можно, используя один резистор, доработать проект так, чтобы он обеспечивал отсчёт до 19 секунд?

5 Лабораторная работа №5. Вывод данных через последовательный интерфейс

5.2 Введение

UART и USART

В последовательном интерфейсе для передачи данных в одном направлении используется одна сигнальная линия, по которой информационные биты передаются друг за другом — последовательно. Последовательная передача позволяет сократить количество сигнальных линий и добиться улучшения связи на больших расстояниях.

Универсальный асинхронный приёмопередатчик (Universal Asynchronous Receiver-Transmitter) — это специальное устройство, предназначенное для организации связи одной ЭВМ с другими вычислительными устройствами. Оно преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии. Метод преобразования стандартизован и широко применяется в компьютерной технике, особенно во встраиваемых устройствах и системах на кристалле (SoC).

UART представляет собой логическую схему, с одной стороны подключённую к шине вычислительного устройства, а с другой — имеющую два или более выводов для внешнего соединения.

USART (Universal Synchronous-Asynchronous Receiver/Transmitter) — универсальный синхронно-асинхронный приёмопередатчик — аналогичный UART интерфейс, но дополнительно к возможностям UART поддерживающий режим синхронной передачи данных с использованием дополнительной линии тактового сигнала. Впрочем, синхронная передача используется гораздо реже, чем асинхронная.

UART может представлять собой отдельную микросхему или являться частью большой интегральной схемы. Он используется для передачи данных через последовательный порт компьютера и часто встраивается в микроконтроллеры. Из семейства UART наиболее известен протокол RS-232 (COM-порт некоторых ПЭВМ и промышленных ЭВМ). Он дожил до наших дней и не потерял своей актуальности.

COM-порт (интерфейс стандарта RS-232)

COM-порт — интерфейс ЭВМ, соответствующий стандарту EIA/TIA-232E. В случае его отсутствия можно использовать переходник USB-RS-232, например, MOXA 1110.

Данный интерфейс носит название RS-232 и является стандартом физического уровня для интерфейса UART. Он определяет набор используемых

сигнальных линий и уровни для сигналов. Используемые уровни сильно отличаются от традиционных ТТЛ или КМОП-уровней. Во-первых, используются двухполярные сигналы, во-вторых, сигнал положительной полярности соответствует логическому нулю. Для согласования уровней сигналов интерфейса RS-232 и остальной части схемы используются специализированные микросхемы преобразования уровней.

Для сигналов (таблица 3) используются следующие уровни.

Для драйвера (выход):

+5...+12 В — логический нуль (SPACE);

-5...-12 В — логическая единица (MARK).

Вход должен иметь сопротивление в пределах 3..7 кОм и должен быть рассчитан на сигналы:

+3...+25 В — логический нуль;

-3...-25 В — логическая единица.

Также любой вывод интерфейса должен выдерживать замыкание на любой другой вывод и на источник напряжения +5 В. В таблице 4 указаны обозначения для сигналов, принятые для COM-порта, обозначения в соответствии с RS-232, номера выводов в разъёмах и краткое описание назначения сигналов.

Таблица 4. Сигналы интерфейса RS-232

COM	RS-232	DB-9M	DB-25M	I /O	Назначение
PG	AA	5	1	-	Protective Ground - защитная земля, соединяется с корпусом прибора.
SG	AB	5	7	-	Signal Ground - общий провод для сигнальных линий.
TD	BA	3	2	O	Transmit data - передача данных из порта.
RD	BB	2	3	I	Receive Data - прием данных в порт.
RTS	CA	7	4	O	Request to send - запрос COM-порта на передачу данных (сигнал COM-порта о готовности принимать данные).
CTS	CB	8	5	I	Clear to send - вход для разрешения COM-порту передавать данные.
DSR	CC	6	6	I	Data set ready - вход сигнала готовности от подключенного к порту устройства.
DTR	CD	4	20	O	Data terminal ready - сигнал готовности COM-

					порта к обмену данными.
CD	CF	1	8	I	Carrier Detected - сигнал обнаружения несущей (от модема).
RI	CE	9	22	I	Ring indicator - сигнал от модема о получении звонка.

Для подключения к интерфейсу используются 25-контактные или 9-контактные разъёмы (DB25, DB9). Первоначально применялись 25-контактные разъёмы, но многие сигналы устройствами не использовались. В связи с этим произошёл переход к девятиконтактным разъёмам. В оконечном оборудовании используются разъёмы типа DB-9M. В аппаратуре передачи данных (например, стойки ЧПУ) используются разъёмы типа DB-9F.

При асинхронной передаче каждому байту предшествует старт-бит, сигнализирующий приемнику о начале посылки, за которым следуют биты данных и, возможно, бит четности. Завершает посылку стоп-бит, гарантирующий паузу между посылками (рисунок 32). Старт-бит следующего байта посылается в любой момент после стоп-бита, то есть между передачами возможны паузы произвольной длительности. Старт-бит, имеющий всегда строго определенное значение (логический нуль), обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. При этом приемник и передатчик должны работать на одной скорости обмена.

Рисунок 32 – Формат асинхронной передачи

Формат асинхронной посылки позволяет выявлять возможные ошибки передачи: ложный старт-бит, потерянный стоп-бит, ошибку четности. Контроль формата позволяет обнаруживать обрыв линии: при этом принимаются логический нуль, который сначала трактуется как старт-бит, и нулевые биты данных, а потом срабатывает контроль стоп-бита.

Для асинхронного режима принят ряд стандартных скоростей обмена: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200, 38 400, 57 600 и 115 200 бит/с. Количество битов данпых может составлять 5, 6, 7 или 8 (5- и 6-битные форматы применяются редко). Количество стоп-битов может быть 1, 1,5 или 2 («полтора бита» означает только длительность стопового интервала).

Асинхронный режим является байт-ориентированным (символьно-ориентированным): минимальная пересылаемая единица информации — байт (символ). В отличие от него, синхронный режим (не поддерживаемый COM-портами ЭВМ) является бит-ориентированным — кадр, пересылаемый по нему, может иметь произвольное количество битов.

Для преобразования сигналов между уровнями ТТЛ/КМОП логики и уровнями RS-232 существуют специализированные микросхемы.

Сложность согласования состоит в том, что стандарт RS-232 работает на уровнях напряжений +12 В...- 12 В, а микросхемы контроллеров UART работают на уровнях 5 В или 3,3 В. Для согласования уровней напряжения существует несколько вариантов схем преобразователей, но самой популярной остаётся микросхема MAX232 и ее аналоги.

Для работы внутреннего умножителя напряжения микросхеме требуется четыре керамических конденсатора номиналом 0,1 мкФ. В ходе настоящей лабораторной работы используется микросхема MAX202. Подключение к ней конденсаторов приведено на рисунке 33.

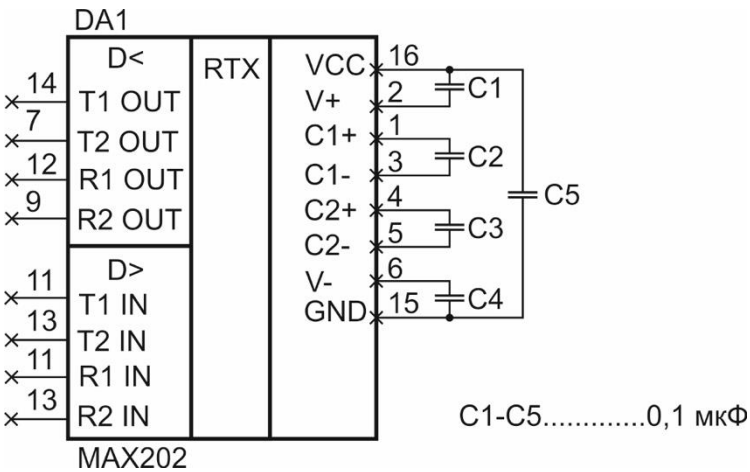


Рисунок 33 – Подключение конденсаторов к микросхеме MAX202

Использование интерфейса RS-232 при работе с микроконтроллером STM32F407VGT6

Микроконтроллер STM32F407VGT6, размещенный на отладочной плате STM32F4 Discovery, содержит в своем составе блок USART. Данный модуль используется для связи МК с внешними устройствами.

USART, как и все остальные модули МК, работает от тактовой частоты микроконтроллера, это необходимо учитывать при настройке скорости обмена. Схема подключения платы STM32F4 Discovery к остальным компонентам схемы рассмотрена ниже.

Для согласования уровней сигналов при подключении МК к ПЭВМ используется микросхема MAX202. Выводы PB6 и PB7 платы STM32F4 Discovery используются для приема и отправки данных (TxD и RxD соответственно) (рисунок 34). При сборке схемы кабель E1 следует подключать в последнюю очередь.

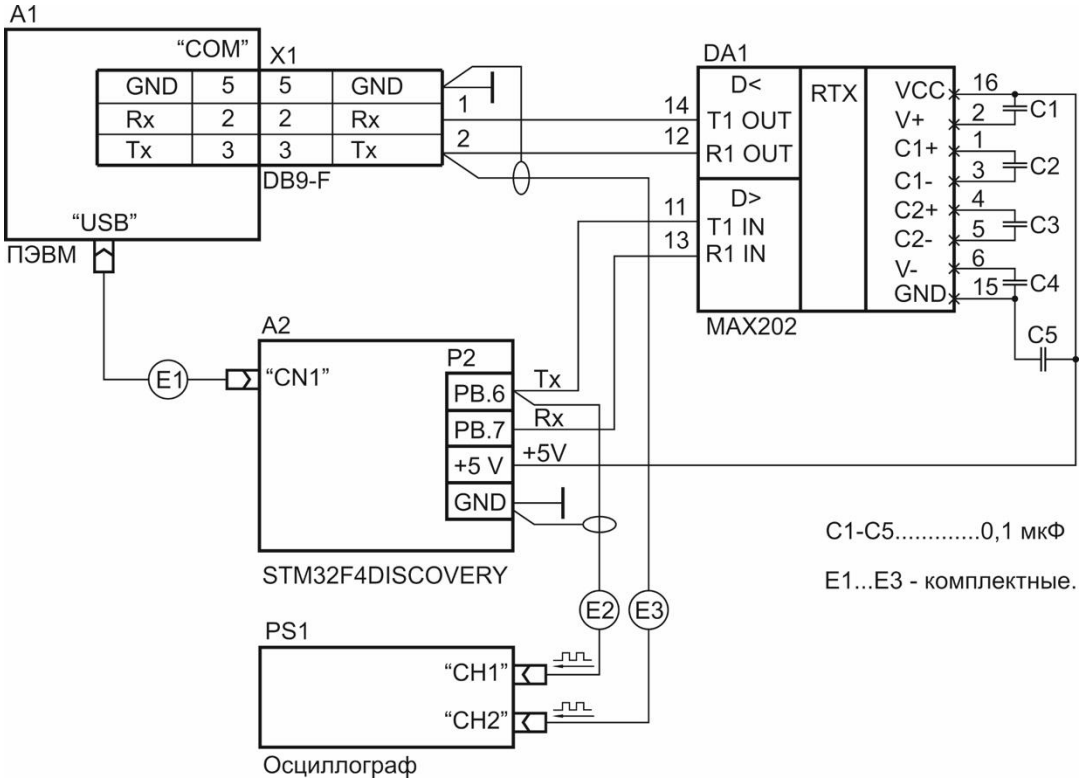


Рисунок 34 – Схема подключения оборудования к плате STM32F4 Discovery

Работа интерфейса USART в контроллерах семейства STM32

USART в микроконтроллерах STM32 предоставляет гибкие средства для полнодуплексного обмена данными с внешними устройствами в последовательном формате с возможностью поддержки сигналов CTS/RTS; поддерживает полудуплексный обмен по однопроводной линии; может работать в широком диапазоне скоростей передачи. В мультибуферном режиме DMA достигается высокая скорость передачи данных и максимальное её значение составляет 3 Мбит/с. Также поддерживается однонаправленная передача в синхронном режиме: мультипроцессорная связь, LIN (local interconnection

network) — сеть для локальной связи, smartcard — протокол, инфракрасный протокол в соответствии со спецификацией IrDA (infrared data association) SIR ENDEC. Структурная схема передатчика модуля USART приведена на рисунке 35.

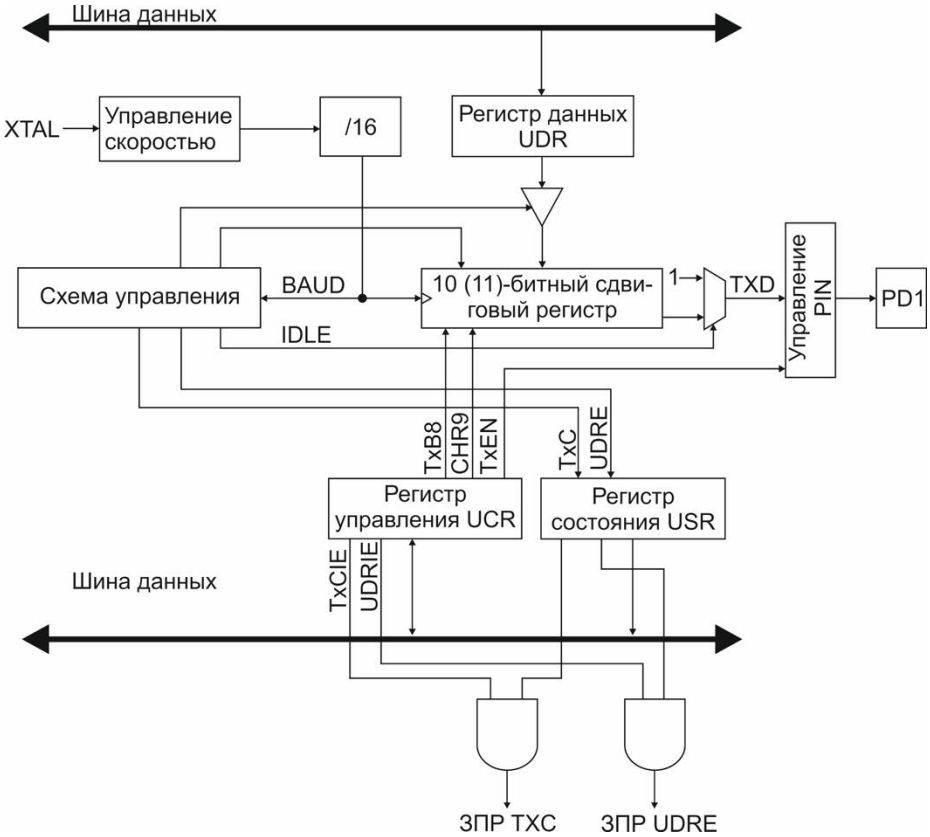


Рисунок 35 – Структурная схема передатчика модуля USART

Работа передатчика разрешается установкой в «1» разряда TXEN регистра UCR. Если этот разряд сброшен (передатчик выключен), вывод PD1 (TXD) может использоваться как линия ввода/вывода порта PD. При установке разряда TXEN этот вывод подключается к передатчику UART и начинает функционировать как выход независимо от состояния 1-го разряда регистра DDRD.

Передача инициируется записью передаваемых данных в регистр данных UDR. После этого данные пересылаются из регистра UDR в сдвиговый регистр передатчика. При этом возможны два варианта:

1) новое значение записывается в регистр UDR после того, как был передан стоп-бит предыдущего слова. В этом случае данные пересылаются в сдвиговый регистр сразу же после записи в регистр UDR;
2) новое значение записывается в регистр UDR во время передачи. В этом случае данные пересылаются в сдвиговый регистр после передачи стоп-бита текущего слова.

После пересылки содержимого регистра UDR в сдвиговый регистр флаг UDRE регистра USR устанавливается в «1», что означает готовность передатчика к получению нового значения. В этом состоянии флаг остаётся до новой записи в регистр UDR. Одновременно с пересылкой формируется служебная информация: нулевой разряд сдвигового регистра сбрасывается в «0» (старт-бит), а девятый (или десятый) разряд устанавливается в «1» (стоп-бит). Если включен режим передачи девятиразрядных данных (разряд CHR9 регистра UCR установлен в «1»), то значение разряда TXB8 регистра UCR копируется в девятый разряд сдвигового регистра.

После загрузки сдвигового регистра его содержимое начинает сдвигаться вправо и поступает на вывод TXD в следующем порядке: стартовый бит, данные (начиная с младшего разряда), стоповый бит. Сдвиг осуществляется по тактовому сигналу, вырабатываемому контроллером скорости передачи. Если во время передачи в регистр UDR было записано новое значение, то после передачи стоп-бита оно пересылается в сдвиговый регистр. Если же к моменту окончания передачи стоп-бита такой записи выполнено не было, в регистре USR устанавливается флаг завершения передачи TXC.

5.3 Описание лабораторной работы

Цели работы

1. Освоить вывод данных через последовательный интерфейс;
2. Освоить приёмы работы при подключении оборудования через последовательный интерфейс;

Указания к выполнению работы

Лабораторная работа состоит из двух частей. В ходе выполнения первой части производится сборка электрической схемы. При выполнении второй части производится программирование контроллера и отладка программы.

Порядок выполнения работы

Часть 1:

1. Выясните местоположение необходимых контактов согласно КД на отладочную плату STM32F4 Discovery;
2. Соберите схему.

Часть 2:

1. Запрограммируйте контроллер;
2. Проконтролируйте выполнение программы.

Требования к отчету

Отчет должен быть оформлен в соответствии с требованиями, представленными в Приложении А.

Отчет должен включать:

1. Титульный лист
2. Цели работы
3. Отчёт о выполнении заданий части 1:
   — Электрическая схема
   — Схема подключения лабораторного оборудования
4. Отчёт о выполнении заданий части 2:
   — Код программы
   — UML диаграмма действий, описывающая алгоритм работы программы
   — Осциллограмма сигнала
   — Вид переданного сообщения (окно с сообщением)
5. Вывод по работе

5.4 Выполнение работы

Для того, чтобы передать информацию по интерфейсу USART, необходимо выполнить следующий алгоритм:

1. Инициализировать USART, для чего:
   - задать используемые контакты контроллера,
   - сконфигурировать используемые контакты контроллера,
   - задать частоту передачи данных,
   - задать количество бит данных,
   - задать количество стоповых бит,
   - задать наличие бита чётности,
   - задать управление потоком.
2. Задать текстовую строку.
3. Подать первый бит текстовой строки в USART.
4. Повторять п. 3 до тех пор, пока символы в строке не закончатся.

Ниже приведен текст программы для передачи данных по USART.

```
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
#include <stm32f4xx.h>
#include <misc.h>
#include <stm32f4xx_usart.h>

#define MAX_STRLEN 12 // максимальная длина передаваемой строки
volatile char received_string[MAX_STRLEN+1]; // переменная для хранения принятой строки
```

```
void Delay(__IO uint32_t nCount) {
  while(nCount--) {
}
}

/* Функция инициализации USART1
 *
 * Аргументы: baudrate --> частота передачи данных по USART
 *
 */

void init_USART1(uint32_t baudrate){

      GPIO_InitTypeDef GPIO_InitStruct; // задание контактов порта, которые
                                        //будут использоваться как TX и RX
      USART_InitTypeDef USART_InitStruct; // для инициализации USART1
      NVIC_InitTypeDef NVIC_InitStructure; // для конфигурирования NVIC (nested
                                           //vector interrupt controller)

      /* Задание тактовой частоты для USART1
       * заметим что только USART1 и USART6 подключены к APB2
       * остальные USART подключены к APB1
       */
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);

      /* задание тактовой частоты для контактов порта, используемых
       * USART1, PB6 в качестве TX и PB7 в качестве RX
       */
      RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);

      /* задание 6 и 7 контактов порта как
       * USART1
       */
      GPIO_InitStruct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
      GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF; // перевод режим работы контактов в
//альтернативный
      GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz; // задание частоты работы порта
      GPIO_InitStruct.GPIO_OType = GPIO_OType_PP; // включение подтягивающих
//резисторов
      GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_UP; // подключение подтягивающих
//резисторов
      GPIO_Init(GPIOB, &GPIO_InitStruct); // инициализация порта с заданными
//параметрами

      // Включение RX и TX в режиме альтернативной функции

          GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_USART1);
          GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_USART1);
      // Задание режима работы

      USART_InitStruct.USART_BaudRate = baudrate; // скорость передачи данных
      USART_InitStruct.USART_WordLength = USART_WordLength_8b;// размер байта: 8 бит
      USART_InitStruct.USART_StopBits = USART_StopBits_1; // один стоповый бит
      USART_InitStruct.USART_Parity = USART_Parity_No; // без бита четности
      USART_InitStruct.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
```

```
// без управления потоком
      USART_InitStruct.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
// включение на прием и передачу
      USART_Init(USART1, &USART_InitStruct); // задание параметров работы USART1

      // Описание прерывания по приёму USART1

      USART_ITConfig(USART1, USART_IT_RXNE, ENABLE); // включение прерывания по приёму
//USART1
      NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn; // конфигурирование канала
//NVIC
      NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;// задание приоритета
//прерывания
      NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // задание субприоритета
//прерывания
      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; // включение прерываний USART1
      NVIC_Init(&NVIC_InitStructure); // инициализация NVIC
      // включение USART1
      USART_Cmd(USART1, ENABLE);

}

/* Данная функция предназначена для отправки строки символов через
 * USART, номер которого задается переменной USARTx.
 *
 * Функция имеет два аргумента: USARTx --> номер, например USART1, USART2 и т.д..
 *                          и (volatile) char *s --> строка, подлежащая отправке
 *
 * Отметим, что передать строку типа string в функцию нельзя,
 *            поскольку действуют ограничения со стороны компилятора.
 *            Поэтому строка передается в виде указателя на массив символов.
 *
 * Также отметим, что указатель объявляется типа volatile char, поскольку переменная
 * received_string объявлена типа  volatile char, а это сделано в свою очередь потому,
 * что иначе компилятор может оптимизировать её, в результате чего программа работать
 * не будет
 */

void USART_puts(USART_TypeDef* USARTx, volatile char *s){

      while(*s){
            // до тех пор, пока не опустошится регистр данных
            while( !(USARTx->SR & 0x00000040) );
            USART_SendData(USARTx, *s);
            *s++;

      }

}

int main(void) {

  init_USART1(9600); // инициализируем USART1 на скорости 9600 бит/с

  USART_puts(USART1, "1"); // отправка сообщения
```

```
  while (1){

        USART_puts(USART1, "1");

    }

}

// это обработчик прерывания (IRQ) для всех прерываний USART1

void USART1_IRQHandler(void){

      // В случае, если выставлен флаг прерывания по приёму USART1
      if( USART_GetITStatus(USART1, USART_IT_RXNE) ){

            static uint8_t cnt = 0; // эта переменная хранит длину строки
            char t = USART1->DR; // переменная для хранения символа из регистра данных
//USART1

            /* Проверить, не является ли принятый символ символом конца строки
             * и не достигнута ли максимальная длина строки
             */
            if( (t != '\n') && (cnt < MAX_STRLEN) ){
                  received_string[cnt] = t;
                  cnt++;
            }
            else{ // В противном случае обнулить счётчик символов и вывести принятую
строку
                  cnt = 0;
                  USART_puts(USART1, received_string);
            }
      }
}
```

Отметим, что для корректной компиляции программы необходимо добавить в директорию проекта файлы «stm32f4xx_tim.h» и «stm32f4xx_tim.c», «stm32f4xx_usart.c», «stm32f4xx_usart.h», «misc.c» и «misc.h», предварительно загруженные из Интернет. Затем файлы копируются в папку cmsis (рисунок 36).

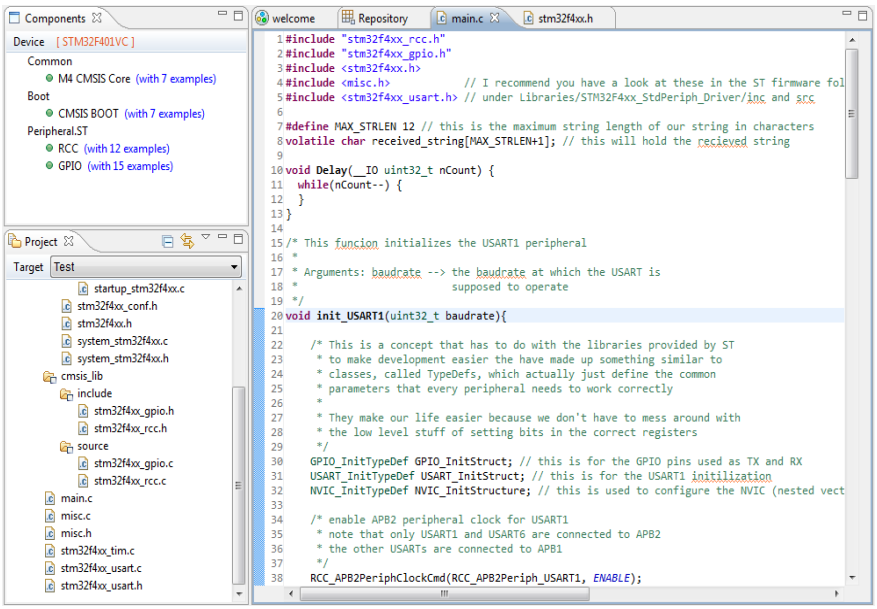


Рисунок 36 – (пояснение в тексте)

Для того, чтобы добавить файл, нажатием правой кнопки мыши на директории проекта следует выбирать пункт «Add Files», затем в появившемся окне выбирать файл, файл например, stm32f4xx_tim.h. Аналогичным образом добавляются файлы «stm32f4xx_usart.c», «stm32f4xx_usart.h», «misc.c» и «misc.h».

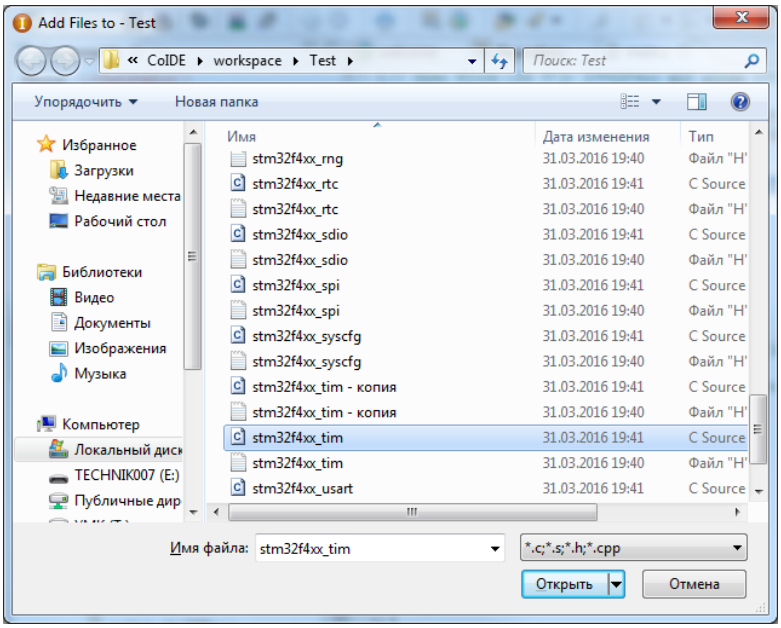


Рисунок 37 – Добавление файла в проект

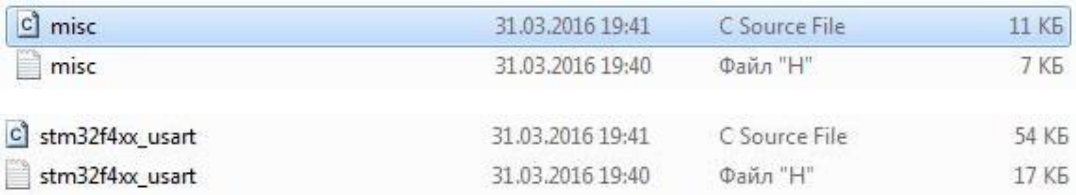


Рисунок 38 – Добавленые в проект файлы

Далее с помощью осциллографа следует произвести контроль первого канала. Изображение должно быть аналогичным приведенному на рисунке 39.

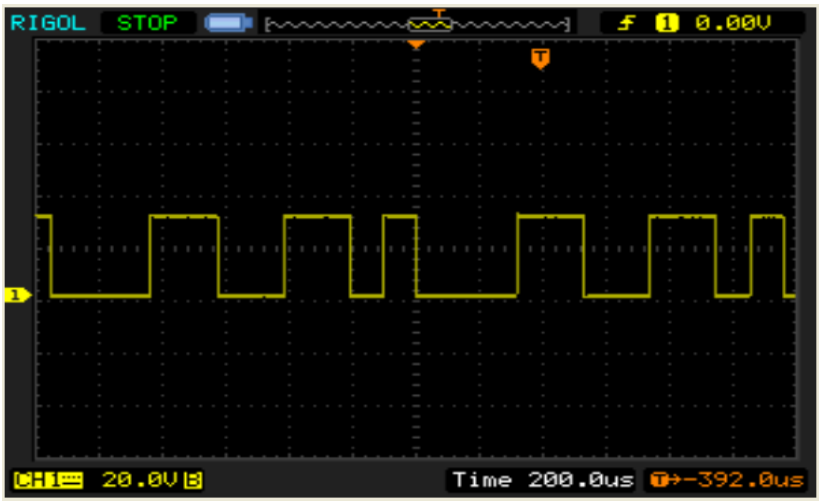


Рисунок 39 – Осциллограмма сигнала до преобразователя

Затем с помощью осциллографа следует произвести контроль второго канала. Изображение должно быть аналогичным приведенному на рисунке 40.

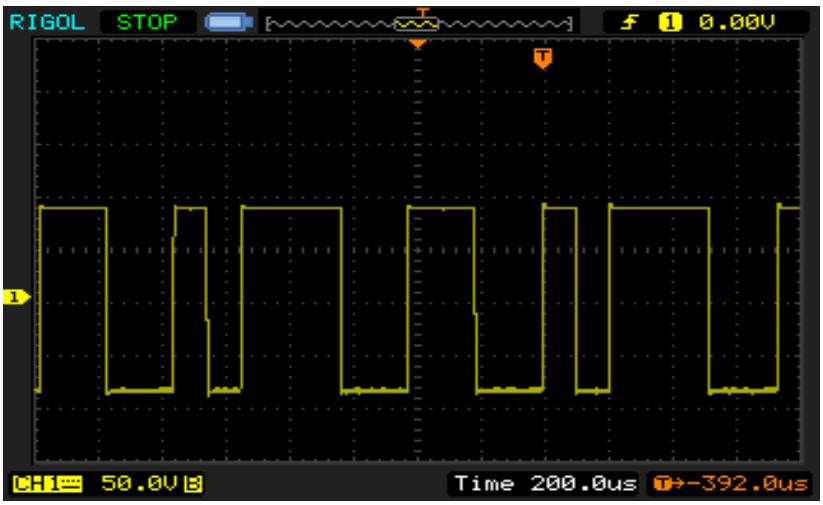


Рисунок 40 – Осциллограмма сигнала после преобразователя

После запуска программы на плате STM32F4 Discovery на ПЭВМ необходимо выполнить следующее:

1. Нажать сочетание клавиш «WinKey+Break», в появившемся окне выбрать вкладку «Оборудование» (рисунок 41).

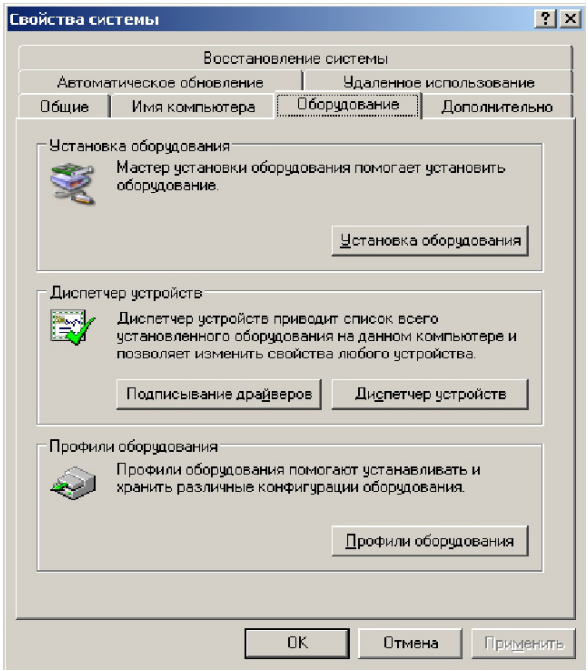


Рисунок 41 – Вкладка «Оборудование»

2. Далее нажать кнопку «Диспетчер устройств» и во вкладке «Порты COM и LPT» уточнить номер COM-порта, к которому подключена плата (рисунок 42).

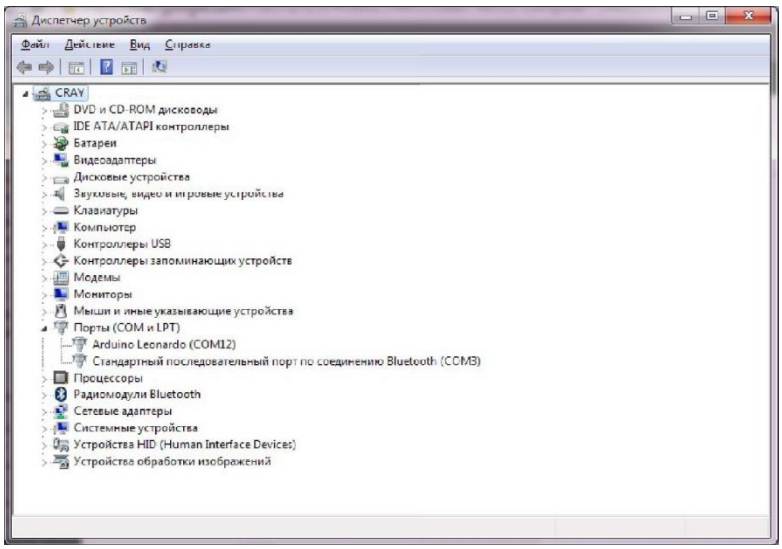


Рисунок 42 – Окно «Диспетчер устройств»

3. Запустить программу «Putty.exe». В появившемся окне выбрать «Serial» и указать заранее уточненный номер порта (п. 2) в окне «Serial line». В окне «Speed» должно быть указано значение «9600» (рисунок 43).

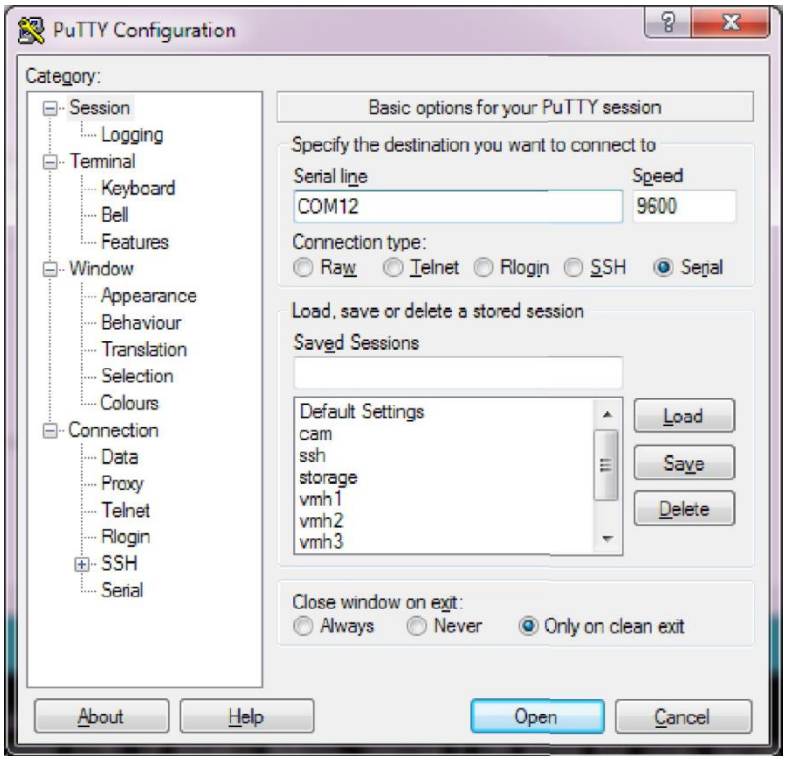


Рисунок 43 – Настройки программы «Putty.exe»

Если все сделано правильно, то в окне программы «Putty.exe» появятся бесконечные строки, состоящие из символа единицы (рисунок 44).

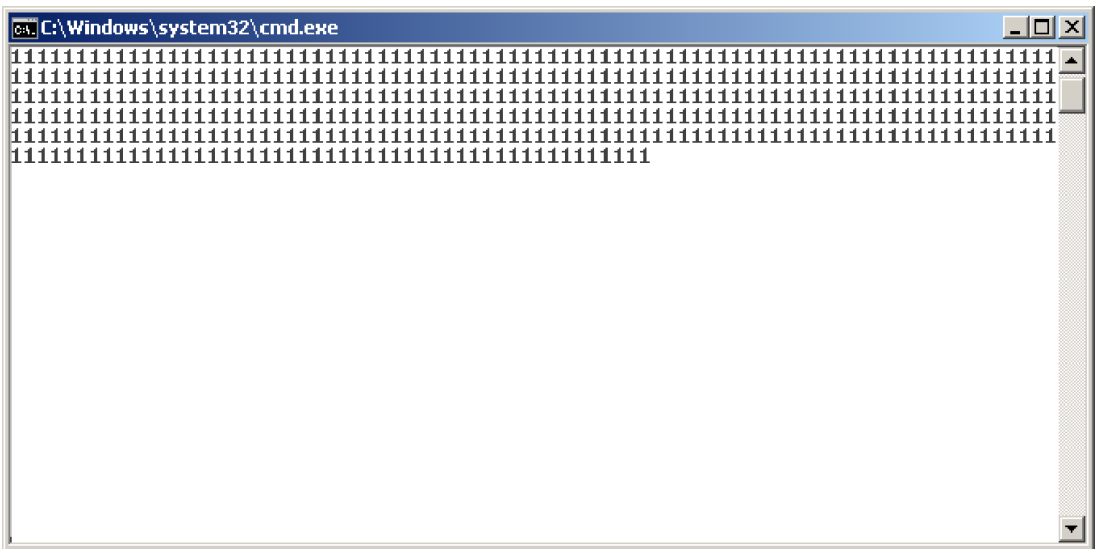


Рисунок 44 – Работа ПЭВМ в качестве приёмного устройства

Контрольные вопросы:

1. Что такое UART?
2. В чем отличие UART и USART?
3. Что такое RS-232?
4. Какие настройки необходимо задавать при конфигурации USART?

Каким образом необходимо изменить текст программы, чтобы вместо символа «1» она отправляла несколько символов?

Список литературы

1. Сайт компании ST Microelectronics [Электронный ресурс] : сохранено в 13:19 UTC 26 февраля 2020 / ST Microelectronics // ST Microelectronics. — Электрон. дан. — ST Microelectronics, 2020 — Режим доступа: https://www.st.com/content/st_com/en.html.

2. Discovery kit with STM32F407VG MCU [Электронный ресурс] : сохранено в 13:19 UTC 26 февраля 2020 / ST Microelectronics // ST Microelectronics. — Электрон. дан. — ST Microelectronics, 2020 — Режим доступа: https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-discovery-kits/stm32f4discovery.html.

3. CoIDE [Электронный ресурс] : сохранено в 13:19 UTC 26 февраля 2020 / ST Microelectronics // ST Microelectronics. — Электрон. дан. — ST Microelectronics, 2020 — Режим доступа: https://www.st.com/en/development-tools/coide.html.

4. GNU Arm Embedded Toolchain. Pre-built GNU toolchain for Arm Cortex-M and Cortex-R processors [Электронный ресурс] : сохранено в 13:21 UTC 26 февраля 2020 / Arm Limited // Arm Limited. — Электрон. дан. — Arm Limited, 2020 — Режим доступа: https://developer.arm.com/tools-and-software/open-source software/developer-tools/gnu-toolchain/gnu-rm.

5. Компилятор GNU Tools for ARM Embedded Processors сохранено в 13:28 UTC 26 февраля 2020 / http://firsthand.ru // http://firsthand.ru. — Электрон. дан. — http://firsthand.ru, 2020 — Режим доступа: http://firsthand.ru/book/programmirovanie/kompilyator-gnu-tools-for-arm-embedded-processors.

6. ST-LINK/V2 [Электронный ресурс] : сохранено в 13:39 UTC 26 февраля 2020 / ST Microelectronics // ST Microelectronics. — Электрон. дан. — ST

Microelectronics, 2020 — Режим доступа: https://www.st.com/en/development-tools/st-link-v2#tools-software.

7. Оценочные и отладочные платы на основе микроконтроллера [Электронный ресурс] : сохранено в 14:50 UTC 26 февраля 2020 / ЗАО «ЧИП и ДИП» // ЗАО «ЧИП и ДИП». — Электрон. дан. — ЗАО «ЧИП и ДИП», 2020 — Режим доступа: https://www.chipdip.ru/catalog-show/evaluation-boards-on microcontroller?gq=stm32f4Discovery.

8. Программирование STM32F4 [Электронный ресурс] : сохранено в 15:00 UTC 26 февраля 2020 / ДРУиД // ДРУиД. — Электрон. дан. — ДРУиД, 2020 — Режим доступа: https://druid.su/rubrik-stm32_program-27.html/.

9. STM32 простой и быстрый старт с CooCox CoIDE [Электронный ресурс] : сохранено в 15:00 UTC 26 февраля 2020 / Сайт-ПАЯЛЬНИК 'cxem.net' // Сайт-ПАЯЛЬНИК 'cxem.net'. — Электрон. дан. — Сайт-ПАЯЛЬНИК 'cxem.net', 2020 — Режим доступа: http://cxem.net/mc/mc172.php.

10. STM32 CooCox Installation [Электронный ресурс] : Материал из Mikrocontroller.net — свободной энциклопедии, версия, сохранённая в 21:50 UTC 6 февраля 2017 / Авторы Mikrocontroller.net // Mikrocontroller.net, свободная энциклопедия. — Электрон. дан. — Сан- Франциско: Фонд Викимедиа, 2020 — Режим доступа: https://www.mikrocontroller.net/articles/STM32_CooCox_Installation.

11. STMicroelectronics. Discovery kit with STM32F407VG MCU User manual : Руководство пользователя / STMicroelectronics, STMicroelectronics, 2017 — 34 с. Текст : электронный // STMicroelectronics : [сайт]. — URL: http://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf (дата обращения: 26.02.2020). — Режим доступа: свободный.

12. STM32F4DISCOVERY, Отладочный комплект на базе STM32F407VGT6 ARM CortexM4-F : Руководство пользователя / STMicroelectronics сохранено в 13:19 UTC 26 февраля 2020 / ST Microelectronics // ST Microelectronics. — Электрон. дан. — ST Microelectronics, 2020 — Режим доступа: Режим доступа : URL: http://www.chipdip.ru/product/stm32f4discovery/. – Загол. с экрана.

13. Применение шифраторов и дешифраторов [Электронный ресурс] : сохранено в 16:10 UTC 26 февраля 2020 / Home Electronics // Home Electronics. — Электрон. дан. — Home Electronics, 2020 — Режим

доступа: http://www.electronicsblog.ru/cifrovaya-sxemotexnika/primenenie shifratorov-i-deshifratorov.html.

14. CooCox CoIDE [Электронный ресурс] : сохранено в 16:00 UTC 26 февраля 2020 / Сайт-ПАЯЛЬНИК 'cxem.net' // Сайт-ПАЯЛЬНИК 'cxem.net'. — Электрон. дан. — Сайт-ПАЯЛЬНИК 'cxem.net', 2020 — Режим доступа: http://cxem.net/software/coide.php.

15. Осваиваем и изучаем микроконтроллер на основе ЦПУ ARM Cortex-M4. [Электронный ресурс] : сохранено в 16:00 UTC 26 февраля 2020 / http://firsthand.ru // http://firsthand.ru. — Электрон. дан. — http://firsthand.ru, 2020 — Режим доступа: http://firsthand.ru/book/osvaivaem-cortex-m4.

16. STM32F405xx STM32F407xx, ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera Datasheet - production data DocID022152 Rev 7 : Руководство пользователя / ST Microelectronics, ST Microelectronics, 2016 — 202 с. — Текст : электронный // Электронно-библиотечная система «Лань» : [сайт]. — URL: www.st.com/resource/datasheet/stm32f405rg (дата обращения: 26.02.2020). —Режим доступа: свободный.

17. Бартенев В. Г., Алгинин Б. Е. От самоделок на логических элементах до микроЭВМ: Кн. для учащихся сред. и ст. шк. возраста.— М.: Просвещение, 1993.— 189 с: ил.— ISBN 5-09-002655-6.

18. ARM Учебный курс. USART [Электронный ресурс] : сохранено в 16:30 UTC 26 февраля 2020 / http://easyelectronics.ru // http://easyelectronics.ru. — Электрон. дан. — http://easyelectronics.ru/tag/uart, 2020 — Режим доступа: http://easyelectronics.ru/tag/uart

19. UART и с чем его едят [Электронный ресурс] : сохранено в 16:31 UTC 26 февраля 2020 / ООО "TechMedia" // ООО "TechMedia". — Электрон. дан. — ООО "TechMedia", 2020 — Режим доступа: https://habr.com/ru/post/109395/

20. UART и USART. COM-порт. Часть 1 [Электронный ресурс] : сохранено в 00:01 UTC 27 января 2016 / hamper // hamper. — Электрон. дан. — hamper, 2020 — Режим доступа: http://www.rotr.info/electronics/mcu/arm_usart.htm

21. MAX232 5В двухканальный приемник/передатчик RS-232 с защитой от электростатического разряда 15 кВ [Электронный ресурс] : сохранено в 17:21 UTC 26 февраля 2020 / ООО "Рынок Микроэлектроники" // ООО "Рынок Микроэлектроники". — Электрон. дан. — ООО "Рынок Микроэлектроники", 2020 — Режим доступа: http://catalog.gaw.ru/index.php?page=component_detail&id=23090.

22. MAX202 5В двухканальный приемник/передатчик RS-232 с защитой от электростатического разряда 15 кВ [Электронный ресурс] : сохранено в 17:21 UTC 26 февраля 2020 / ООО "Рынок Микроэлектроники" // ООО "Рынок Микроэлектроники". — Электрон. дан. — ООО "Рынок Микроэлектроники", 2020 — Режим доступа: http://catalog.gaw.ru/index.php?page=component_detail&id=23084

23. Васюкевич С.Ю. Вычислительные средства радиосистем. Лабораторный практикум: учеб.- метод. пособие / С. Ю. Васюкевич, И. Г. Давыдов, А. В. Цурко. – Минск: БГУИР, 2016. – 128 с. : ил.ISBN 978-985-543-183-2

24. Гук М. Ю. Аппаратные средства 1BM PC. Энциклопедия. 3-е изд. 1072 с.: ил. СПб.: Питер, 2006. - ISBN 5-469-01182-8,

25. MAX202E–MAX213E, MAX232E/MAX241E±15kV ESD-Protected, +5V RS-232 Transceivers Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600, 2005 Maxim Integrated Product

А Требования к оформлению отчетов по лабораторным работам

1. Отчет выполняется в виде самодостаточного документа. Материал, изложенный в отчете, должен быть понятен без дополнительных комментариев со стороны исполнителей.

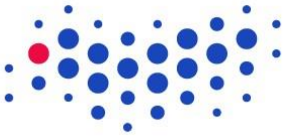
2. Отчет выполняется как текстовый документ в соответствии с ГОСТ 2.105-95.

3. В отчёте должна быть представлена диаграмма действий UML, соответствующая программе.

4. Листы отчета должны быть пронумерованы, кроме титульного листа, который считается первым.

5. Если отчет содержит большое количество листов, рекомендуется добавлять лист с содержанием отчета (разделы и номера листов).

6. В диаграммах UML не допускается чрезмерное усложнение. Допускается использовать в качестве комментариев части текста программы.



Федеральное Государственное Автономное Образовательное Учреждение Высшего Профессионального Образования Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Отчёт по лабораторной работе

Лабораторная работа № ___

Название лабораторной работы: _____

Выполнил: _____

Группа: _____

Преподаватель: _____

Санкт-Петербург

2020

Цель работы: в данном разделе должна быть сформулирована цель работы. Допускается копирование цели работы из методических указаний.

Задачи, решаемые в работе:

1. Задача 1.
2. Задача 2.
3. Задача N.

Теоретическая часть

Структура отчета по лабораторной работе

Отчет по лабораторной работе должен содержать цель и перечень задач, решаемых в работе. Во время защиты лабораторной работы студенту следует ясно и чётко изложить цель работы и основные решённые задачи, а также ответить на дополнительные вопросы, заданные преподавателем в процессе защиты отчёта.

Теоретическая часть отчёта по лабораторной работе должна включать в себя всю необходимую информацию о предметной области, к которой относятся описание работы программных и аппаратных компонентах, используемых при ее выполнении, выдержки из технической документации и т. п.

Данный раздел не является обязательным, однако рекомендуется оставлять его в отчете, так как он может быть использован как план ответа при защите лабораторной работы. В случае, если теоретическая части отчета по лабораторной работе содержит листинги программного кода, таблицы или рисунки, то они должны быть оформлены по правилам, описанным далее в этом разделе.

Часть отчета, описывающая экспериментальную или практическую часть лабораторной работы, является обязательной и не может быть опущена. Данная часть отражается персональные результаты студента, полученные им при выполнении лабораторной работы. Этими результатами являются листинги написанного программного кода, таблицы и изображения, полученные в процессе решения сформулированных задач. Исходные данные, использованные студентом для практической проверки реализованных в ходе практической работы программных компонентов, также должны быть представлены в экспериментальной части. Данные по выполненной лабораторной работе можно предоставить отдельно, в том числе в виде ссылки на Интернет-ресурс. При изучении ранее выполненных работ студентам рекомендуется самостоятельно переписать всю программу в том виде, в каком она была написана, с целью лучшего понимания алгоритма программы. Если студент не завершил цикл выполнения программы, должно быть проведено ее окончательное редактирование. В экспериментальную часть отчета по лабораторной работе

необходимо включить пояснения и комментарии к написанному программному коду (если написание программного кода является одной из задач лабораторной работы).

Выводы должны быть кратко изложены в виде списка и отражать наиболее важные аспекты лабораторной работы. В конце отчета студент должен привести список использованной при подготовке отчета и процитированной литературы (в том числе ссылки на стандарты, руководства пользователя и прочую техническую документацию). Текст отчета по лабораторной работе может являться планом ответа при защите лабораторной работы. Прямое чтение текста не допускается.

Экспериментальная часть

Правила оформления текста

Отчет должен быть представлен в формате PDF. Поля текста могут быть выбраны произвольно, например, 25 мм по всем сторонам печатного листа. Рекомендуется при оформлении отчета использовать шрифт чёрного цвета с засечками (например, DejaVu Serif) высотой 14 пунктов с полуторным интервалом между строками. Обязательным требованием при оформлении текста отчёта по лабораторной работе является его выравнивание по ширине страницы и использование переносов.

На рисунках в отчете могут быть представлены блок-схемы, графики, изображения аппаратных средств разработки, а также прочая графическая информация. Рисунки должны быть чётки и легкочитаемыми. Если рисунок является снимком экрана, то настоятельно рекомендуется сохранять исходное в форматы без сжатия (например, PNG) или копировать снимок экрана из буфера обмена непосредственно в программу, в которой редактируется текст отчета (например, LibreOffice). Графики и диаграммы рекомендуется создавать в векторных форматах. Ниже представлен пример оформления рисунка.

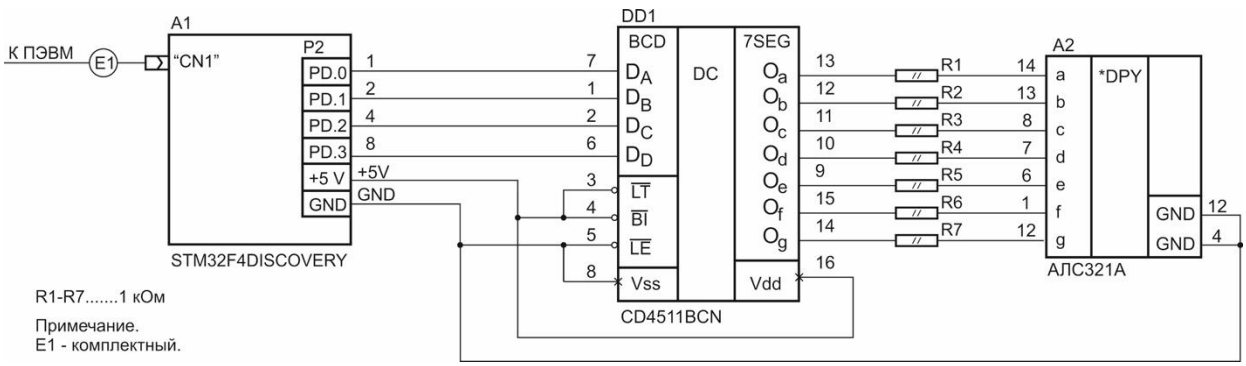


Рисунок А.1 – Пример оформления рисунка

Листинги программного кода должны оформляться как скриншоты окна редактора, либо непосредственно в тексте отчета. В обоих случаях должны быть соблюдены следующие правила:

1. Необходимо использовать моноширинные шрифты.
2. Строки программы должны быть пронумерованы, нумерация должна начинаться с единицы.
3. Должна быть включена «подсветка» синтаксических конструкций используемого языка программирования.

Исходные коды должны быть снабжены комментариями. На рисунки и листинги обязательно должны присутствовать ссылки в тексте. Например "*На рисунке 1 изображен ...* " или *"Исходный код программы представлен в листинге 2 ...*". Листинги, таблицы и рисунки должны быть снабжены подписями, отражающими их содержание.

Выводы

В выводах необходимо кратко пояснить полученные студентом в ходе выполнения лабораторной работы результаты и следующие из них выводы, а также определить, достигнута ли сформулированная цель.

Шубин Антон Сергеевич
Федосов Юрий Валерьевич

Cortex M4. Лабораторный практикум

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверский пр., 49