

Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им Н.Э. Баумана)

Онуфриева Т.А.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
«Интерфейсы периферийных устройств»

Калуга, 2022г.

Лабораторный практикум разработан в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.01 «Информатика и вычислительная техника».

Лабораторный практикум рассмотрен и одобрен:

– кафедрой «Информационные системы и сети» ИУК2
протокол № _____ от « 4 » 02 2022 г.

Заведующий кафедрой ИУК2 _____ к.т.н., доцент И.В. Чухраев

– методической комиссией факультета ИУК
протокол № _____ от « ____ » _____ 2022 г.

Председатель методической комиссии ИУ _____ к.т.н., доц. М.Ю. Адкин

– методической комиссией Калужского филиала МГТУ им. Н.Э. Баумана
протокол № _____ от « ____ » _____ 2022 г.

Председатель методической комиссии Филиала _____ д.э.н., проф. О.Л. Перерва

Рецензент:

к.т.н., доцент кафедры ИУК6

_____ А.Б.Лачихина

Автор: _____ к.т.н., доцент ИУК2 Т.А. Онуфриева

Аннотация

Лабораторный практикум состоит из 4 лабораторных работ и содержит основные материалы для изучения методики проектирования интерфейсов периферийных устройств. Работы выполняются на лабораторном стенде OSUS, разработанном на кафедре, элементная база - современные микроконтроллеры stm32. В практикуме рассмотрены примеры программ по настройке периферийных устройств для данных микроконтроллеров, приведен подробный алгоритм работы с ПО STM32CubeMX. Предназначено для выполнения лабораторных работ по дисциплинам «Интерфейсы периферийных устройств» и «Архитектура ЭВМ».

© Калужский филиал МГТУ им. Н.Э. Баумана, 2022

© Онуфриева Т.А.

Оглавление

ЛАБОРАТОРНАЯ РАБОТА №1.....	3
Структура лабораторного стенда OSUS. ПО STM32CubeMX.....	3
Порядок работы.	4
ЧАСТЬ 1. Основные параметры лабораторного стенда.	4
ЧАСТЬ 2. Создание проекта в среде STM32 CubeIDE.	12
Задание на лабораторную работу:.....	18
Отчет должен содержать	18
Вопросы для самоконтроля	18
ЛАБОРАТОРНАЯ РАБОТА 2.....	19
Программирование дисплея.	19
Порядок работы.	19
Задание на лабораторную работу:.....	23
Отчет должен содержать	23
Вопросы для самоконтроля	23
ЛАБОРАТОРНАЯ РАБОТА 3.....	24
Работа с внешней Flash-памятью.	24
Порядок работы.	24
Задание на лабораторную работу	26
Отчет должен содержать	26
Вопросы для самоконтроля	27
ЛАБОРАТОРНАЯ РАБОТА 4.....	27
Разработка программы обмена данными по интерфейсам.	27
Порядок выполнения работы:	27
Задание на лабораторную работу	27
Отчет должен содержать	27
Вопросы для самоконтроля	27
СПИСОК ЛИТЕРАТУРЫ.....	28

ЛАБОРАТОРНАЯ РАБОТА №1.

Структура лабораторного стенда OSUS. ПО STM32CubeMX.

Цель работы: сформировать навыки прошивки микроконтроллера STM32 с использованием программного обеспечения STM32CubeMX на лабораторном стенде OSUS.

Задачи: Ознакомиться с основными характеристиками и параметрами лабораторного стенда OSUS, разработанным на кафедре «Информационные системы и сети»; ознакомиться с порядком установки программного обеспечения STM32CubeMX. Создать новый проект в соответствии с заданием.

Порядок работы.

ЧАСТЬ 1. Основные параметры лабораторного стенда.

Для успешного выполнения данной лабораторной работы рекомендуется изучить следующую информацию:

Описание стенда OSUS

-Внешний вид лабораторного стенда представлен на рисунке 1

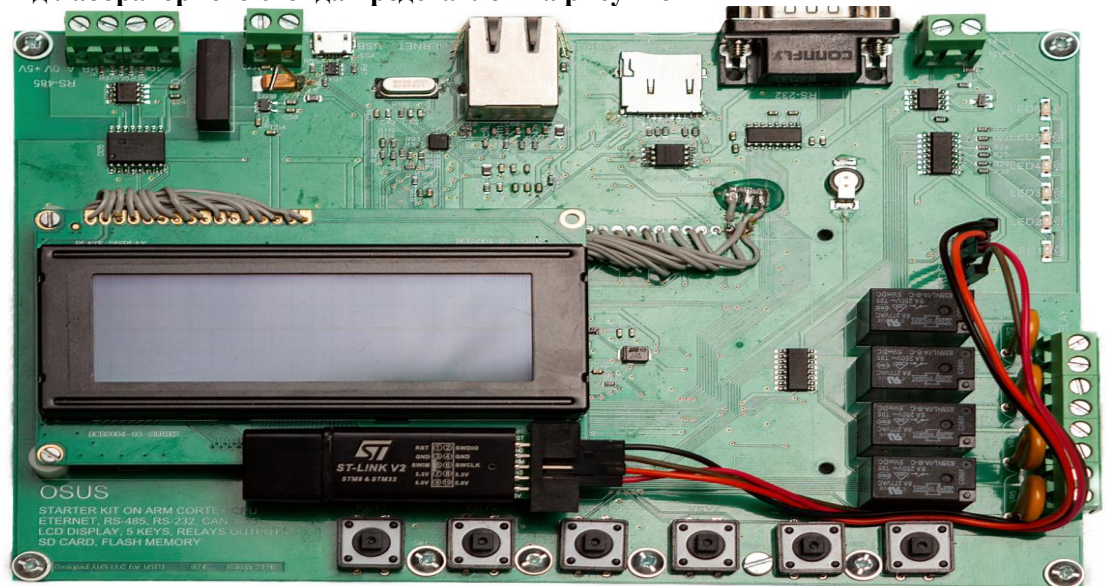


Рисунок 1. Внешний вид стенда OSUS

Основные элементы, расположенные на стенде:

1. 32 разрядный микроконтроллер stm32f407vet6 семейства STM32.
2. 4 реле
3. 6 кнопок
4. Панель индикации, состоящая из 6 светодиодов
5. Питание +5V microUSB – разъем питания платы
6. Преобразователь напряжения 3.3LAN – преобразователь напряжения, из 5В делает 3.3В для питания интерфейса Ethernet
7. Преобразователь напряжения 3.3D - преобразователь напряжения, из 5В делает 3.3В для питания микроконтроллера.
8. Гальваническая развязка – гальваническая развязка, используется для снижения помех и защиты оборудования от повреждения.
9. Преобразование уровней ADUM485, LAN8720, MAX232, PCA82 – преобразователи логических уровней, служат для коммутации интерфейсов с микроконтроллером, т.к. напряжение логической единицы на микроконтроллере отличается от напряжения логических единиц на интерфейсах.
10. Разъем для карты памяти – разъем для карты памяти типа SD.
11. Транзисторная связка – транзисторная связка реализована микросхемой ULN2003a. В данном стенде использована для управления реле и светодиодами. Структурная схема стенда представлена на рисунке 2.

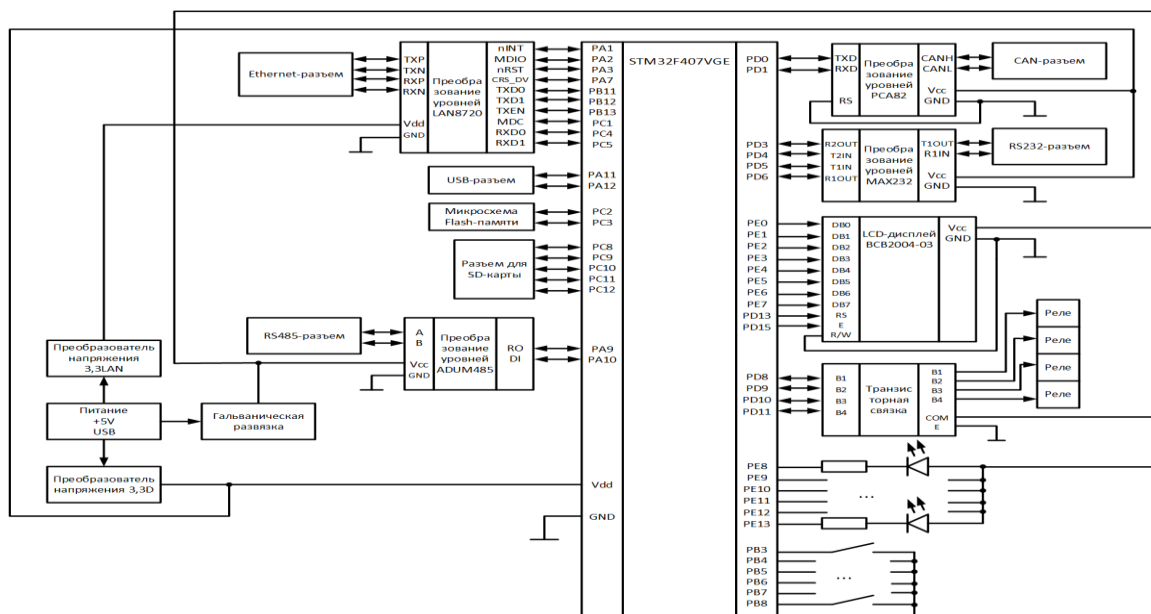


Рисунок 2. Структурная схема лабораторного стенда

Технические характеристики STM32

Фирма STMicroelectronics (STM) одной из первых приступила к серийному выпуску 32-разрядных Flash-микроконтроллеров, в основу которых было заложено ядро ARM Cortex, разработанное специально для встраиваемых применений.

Для разработки учебного стенда был применен микроконтроллер STM32F407VGT6. Семейство STM32F407xx основано на высокопроизводительном 32-разрядном RISC-ядре ARM Cortex-M4, работающем на частоте до 168 МГц. Ядро Cortex-M4 имеет одинарную точность с плавающей запятой (FPU), которая поддерживает все инструкции обработки данных ARM с одинарной точностью и типы данных. Он также реализует полный набор инструкций DSP и блок защиты памяти (MPU), который повышает безопасность приложений. Далее в данной курсовой ядро Cortex-M4 с FPU будет упоминаться как Cortex-M4F.

Семейство STM32F407xx включает в себя высокоскоростную встроенную память (флэш-память до 1 Мбайт, до 192 Кбайт SRAM), до 4 Кбайт резервной памяти SRAM, а также широкий спектр расширенных входов / выходов и периферийных устройств, подключенных к двум APB шинам, трем шинам AHB и 32-битной матрице с несколькими шинами AHB.

APB - шина периферии, предназначенная для организации интерфейса с встроенными периферийными устройствами общего назначения, такими как таймеры, контроллеры прерываний, UART, порты I/O и т.п., и дополнительными периферийными устройствами. С основной системной шиной шина периферии соединяется мостом, обеспечивающим разгрузку системной шины и снижающим общее потребление системы.

Все устройства STM32F407xx имеют три 12-разрядных АЦП, два ЦАП, RTC с низким энергопотреблением, двенадцать 16-разрядных таймеров общего назначения, включая два таймера ШИМ, два 32-разрядных таймера общего назначения и генератор случайных чисел. Они также имеют стандартные и усовершенствованные интерфейсы связи.

Семейство STM32F407xx работает в диапазоне температур от -40 до +105°C от источника питания от 1,8 до 3,6 В. Напряжение питания может упасть до 1,7 В, когда устройство работает в диапазоне температур от 0 до 70°C с использованием внешнего контроллера питания.

Эти особенности делают семейство микроконтроллеров STM32F407xx пригодным для широкого спектра применений:

- Контроль автомобильного двигателя
- Медицинское оборудование
- Промышленное применение: инверторы, автоматические выключатели
- Принтеры и сканеры
- Системы сигнализации, видеодомофон
- Бытовая аудио техника

Принципиальная схема учебного стенда представлена на рис. 4. И включает блоки интерфейсов, реле и светодиодов, питания.

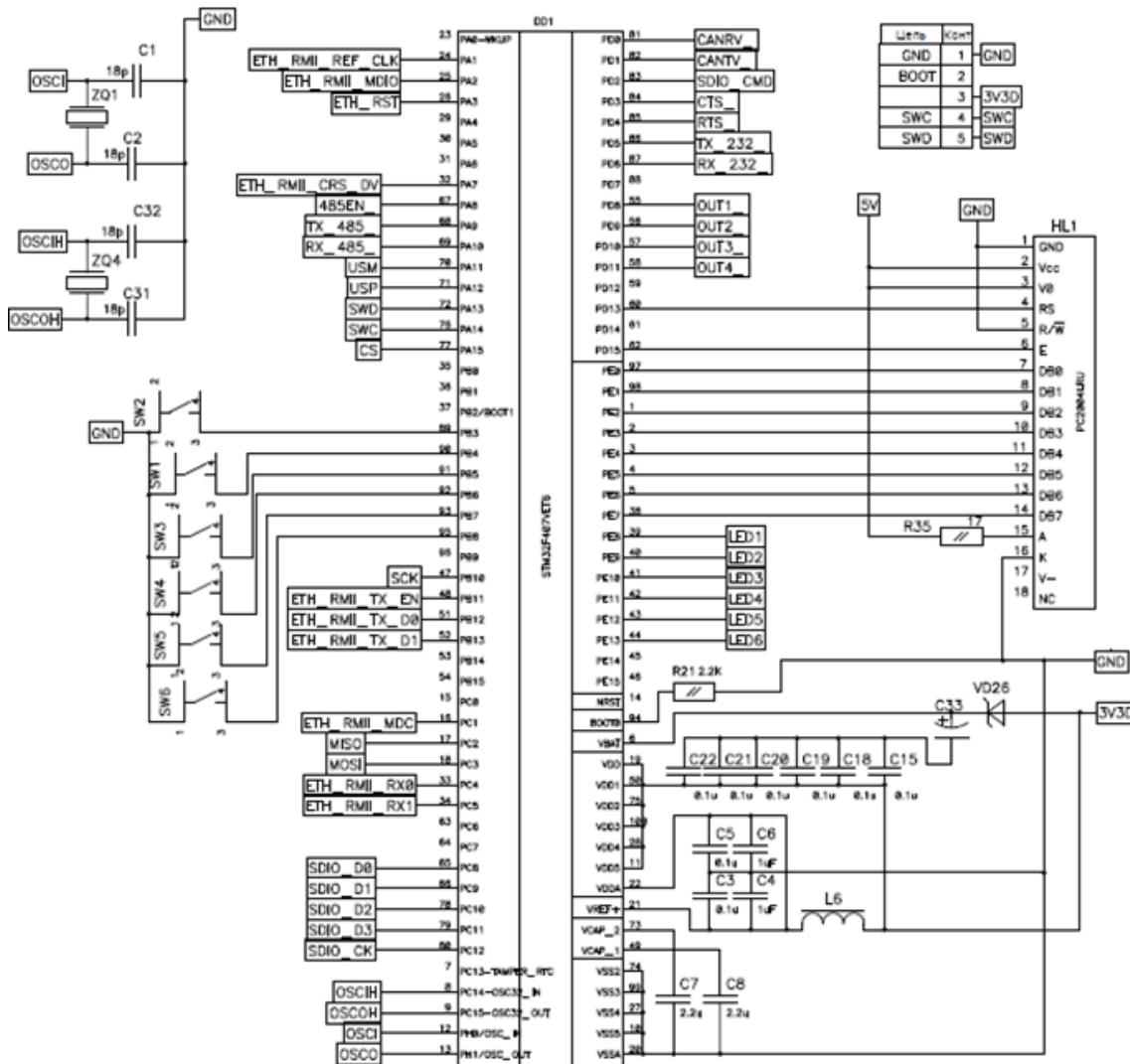


Рисунок 3 – Принципиальная схема платы учебного стенда

Из рисунка видно, что к микроконтроллеру подключен дисплей HL1. Данный дисплей питается от 5 вольт. Также на схеме имеются кварцевые резонаторы zq1 с генерируемой частотой 8 МГц и zq4 с частотой 32768 кГц. Данные резонаторы задают сигнал тактирования. В разработанной схеме используется организация питания 3v3d, данная организация необходима для стабильной работы микроконтроллера. Использование конденсаторов при кварцевых резонаторах также необходимо для стабильной работы, таким образом обеспечивается защита от помех.

Для ввода информации в микроконтроллер служат кнопки. Особенность подключения кнопок заключается в использовании схемы с подтягивающим резистором. Схема подключения контакта с подтягивающим резистором гарантирует высокий логический уровень на входе порта при разомкнутом контакте и низкий логический уровень при замкнутом контакте. Когда контакт разомкнут, вывод микроконтроллера соединен с положительным полюсом источника питания. После замыкания контакта вывод соединяется с «землей», а подтягивающий резистор ограничивает ток, протекающий между полюсами источника питания. Кнопки обозначены на рисунке символами SW1...SW6.

Реализация интерфейсов

Рассмотрим организацию описанной ранее периферии (интерфейсов) на микроконтроллере STM32. Периферийные устройства, описанные в исследовательской части, разрабатывались с помощью справочных листов информации на используемый в данной работе микроконтроллер. На рисунке 4 представлена реализация интерфейса RS-485.

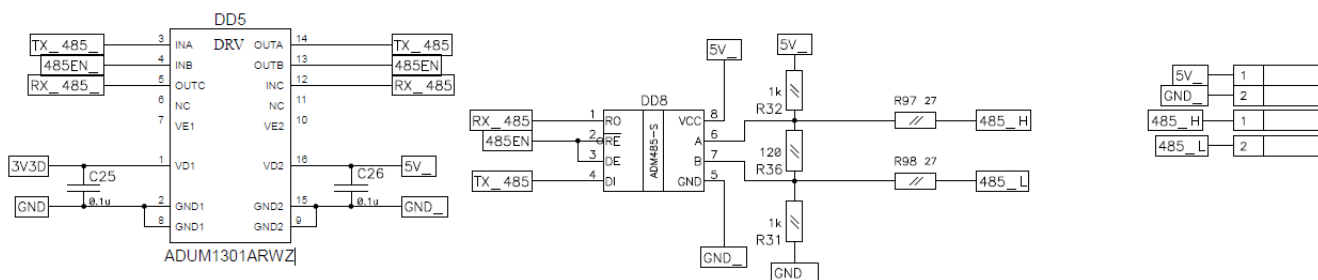


Рисунок 4 – Реализация 485 интерфейса

В данном микроконтроллере присутствует периферийное устройство, называемое **Асинхронный Прием/Передачик** или **UART**. **UART** имеет три основные линии **RX**, **TX** и **GND**. Линия **RX** предназначена для приема данных в микроконтроллер. Линия **TX** предназначена для отправки данных. А линия **GND** служит для создания потенциала относительно которого перемещаются биты.

Из рисунка видно, что сигналы, выработанные микроконтроллером, подаются на микросхему ADUM1301, после чего в микросхему ADM485-S. У микросхемы ADM485-S два сигнала на выходе RX, TX, с подтягиванием к питанию.

Для разработки USB интерфейса в учебном стенде была использована микросхема USBLC6-2. Реализация USB разъема представлена на рисунке 5.

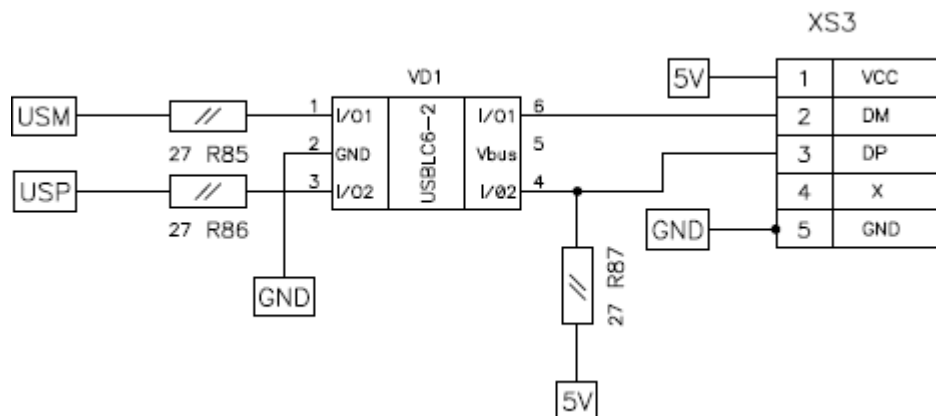


Рисунок 5- Реализация USB разъема

Из рисунка видно, что сигналы USP, USM, вырабатываемые с процессора, поступают на вход микросхемы VD1, после чего выходы микросхемы VD1 соединяются с разъемом XS3, который питается от 5 вольт.

Большинство контроллеров семейства STM32 имеют одно или несколько модулей CAN. Каждый порт - это две ножки: TX - для передачи данных и RX - для приема данных с шины. Реализация CAN- интерфейса представлена на рисунке 6.

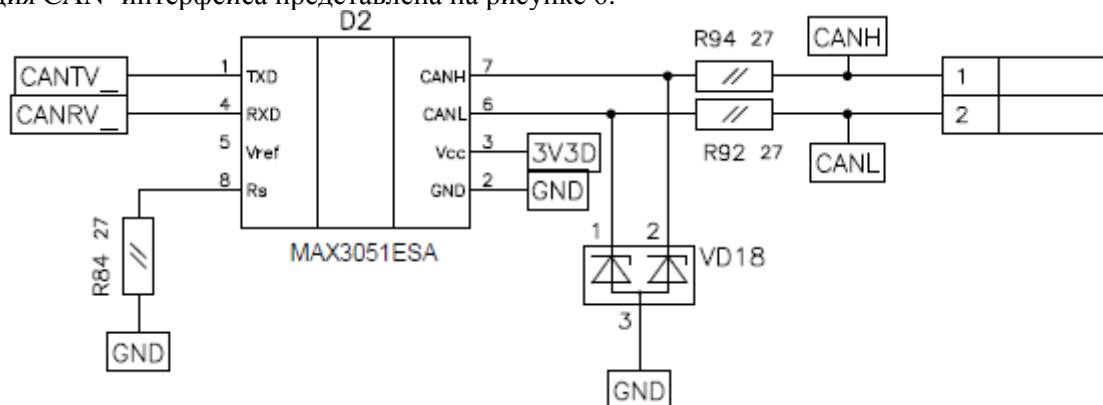


Рисунок 6 – Реализация CAN- шины

Для связи микроконтроллера с шиной недостаточно настроить ножки и соединить их с CAN. Для подсоединения контроллера к шине требуется еще один электронный компонент - приемопередатчик физического уровня сети (трансивер), трансивер - это 8-ми контактная микросхема в корпусе SOIC-8 или DIP-8.

Трансивер в виде отдельной микросхемы является необходимостью. Объясняется это, прежде всего, высокими требованиями к его надежности и рабочим характеристикам, поскольку работает он с цепями, физически выходящими за пределы устройства (в данном случае это сама шина CAN). А условия, в которых находятся эти цепи, зачастую не определены: например, сильные магнитные поля или пролегающие рядом силовые высоковольтные цепи.

Так как STM32 не имеет энергонезависимой EEPROM памяти, поэтому существует следующая проблема. При отключении микроконтроллера от питания заданные инструкции не будут выполняться, настройки и данные не будут сохраняться. Для этого в учебный стенд необходимо добавить внешнюю Flash память. Микросхема внешней памяти и ее реализация представлены на рисунке 7.

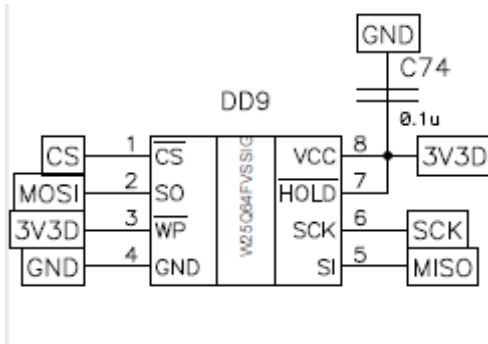


Рисунок 7 – Реализация внешней программной памяти

Следует отметить, что подключение Flash памяти к микроконтроллеру осуществлено с использованием SPI интерфейса.

MAX232 — [интегральная схема](#), преобразующая сигналы последовательного порта [RS-232](#) в сигналы, пригодные для использования в цифровых схемах на базе [TTL](#) или [КМОП](#) технологий. MAX232 работает приемопередатчиком и преобразует сигналы RX, TX, CTS и RTS. Соединение данной интегральной схемы с микроконтроллером представлено на рисунке 8.

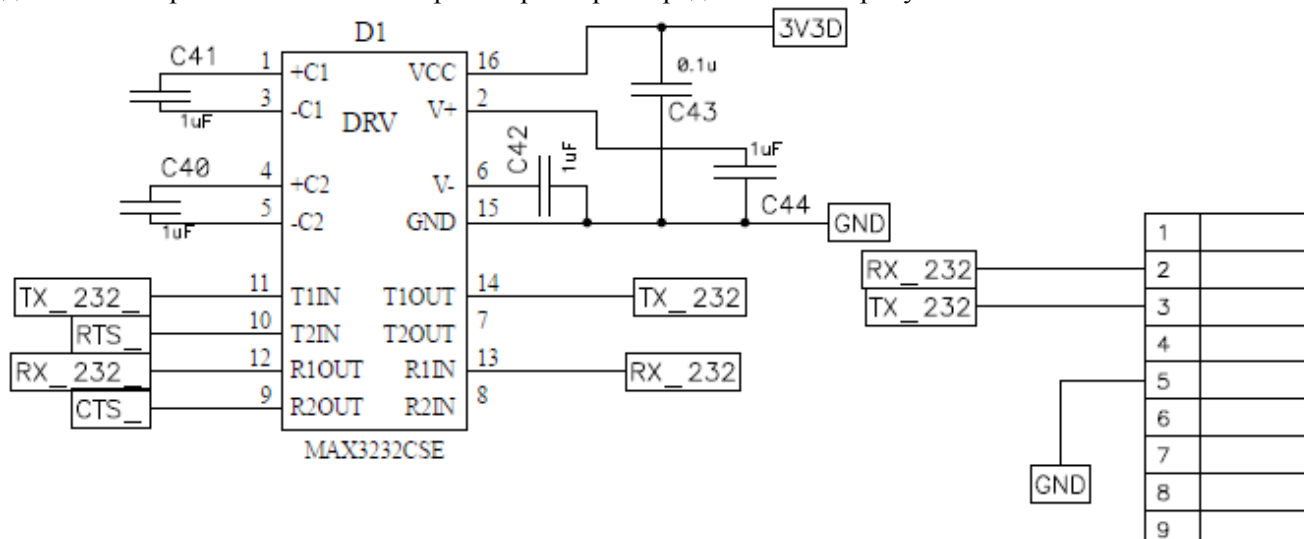


Рисунок 8 – Реализация RS232-интерфейса

Из рисунка видно, что микроконтроллер передает/принимает данные в/из платы через последовательное UART соединение (сигналы TX/RX). Схема обеспечивает уровень выходного напряжения, используемый в RS-232 (приблизительно ± 7.5 В), преобразуя входное напряжение + 5 В при помощи внутреннего зарядового насоса на внешних конденсаторах. Это упрощает реализацию RS-232, так как не требуется усложнять [источник питания](#) только для того, чтобы использовать RS-232.

Для получения или вывода большого объема данных, в учебном стенде необходимо реализовать вместительное хранилище информации. Для этого необходимо добавить разъем под SD карту памяти. Это – объемный носитель информации, по сравнению с обычными микросхемами памяти, объём которых редко превышает 64Мбита. Реализация разъема для карты памяти и sim карты представлена на рисунке 9.

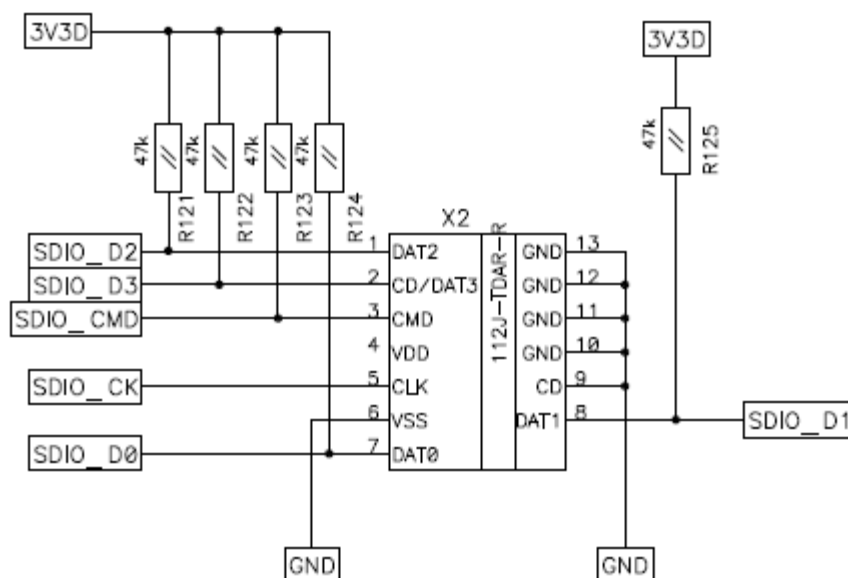


Рисунок 9 – Разъем для карты памяти

Из рисунка видно, что при реализации данного разъема в схему были добавлены подтягивающие резисторы на 47 кОм. Данные резисторы необходимы для стабильной работы.

В данной курсовой работе использовался высокопроизводительный Ethernet модуль LAN8720 ETH Board. Реализация данного модуля в учебном стенде представлена на рисунке 10.

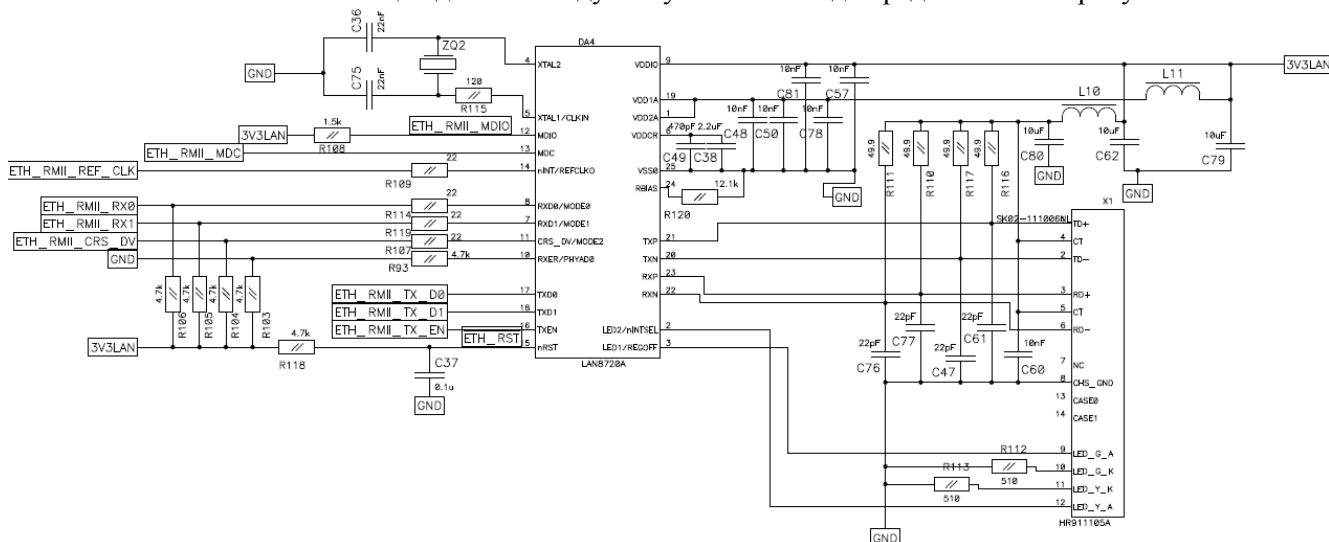


Рисунок 10 – Реализация Ethernet

LAN8720 ETH Board - модуль Ethernet на основе 10/100 трансивера PHY LAN8720 в корпусе QFN-24 (4 x 4 мм) от компании SMSC, являющейся сегодня частью Microchip. Для работы изделия требуется единственный источник питания 3.3 В. Модуль поддерживает интерфейс RMI (Reduce Media Independent Interface) с сокращенным количеством выводов, а также HPAuto-MDIX (Media Dependent Interface). Изделие характеризует гибкая архитектура управления питанием и интегрированный стабилизатор напряжения 1.2 В. Диапазон напряжений I/O: 1.2 В – 3.6 В.

Блок реле и светодиодов

С точки зрения микроконтроллера, реле является мощной индуктивной нагрузкой. Поэтому для включения или выключения реле необходимо использовать транзисторный ключ. Схема подключения представлена на рисунке 11.

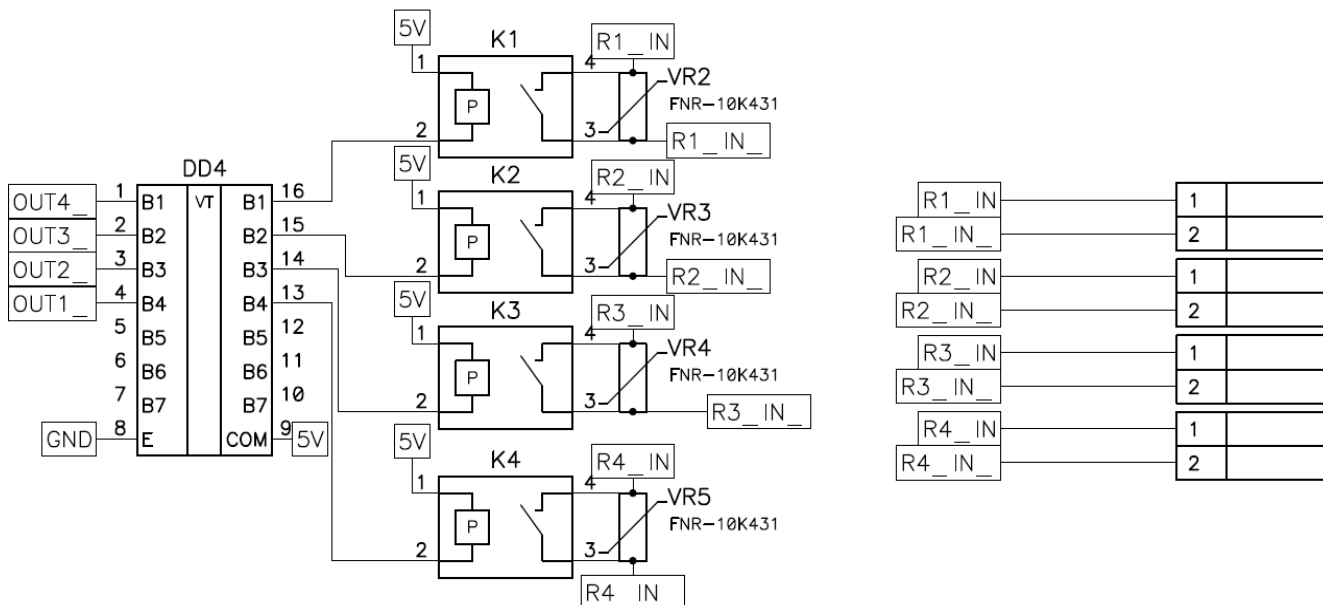


Рисунок 11 – Транзисторная связка

На данном рисунке использованы следующие обозначения: K1-K4 – реле общего назначения, R1-Rn – клеммы.

Светодиоды подключены к микроконтроллеру через транзисторную связку и питаются 5 вольтами. Схема подключения светодиодов к микроконтроллеру представлена на рисунке 12.

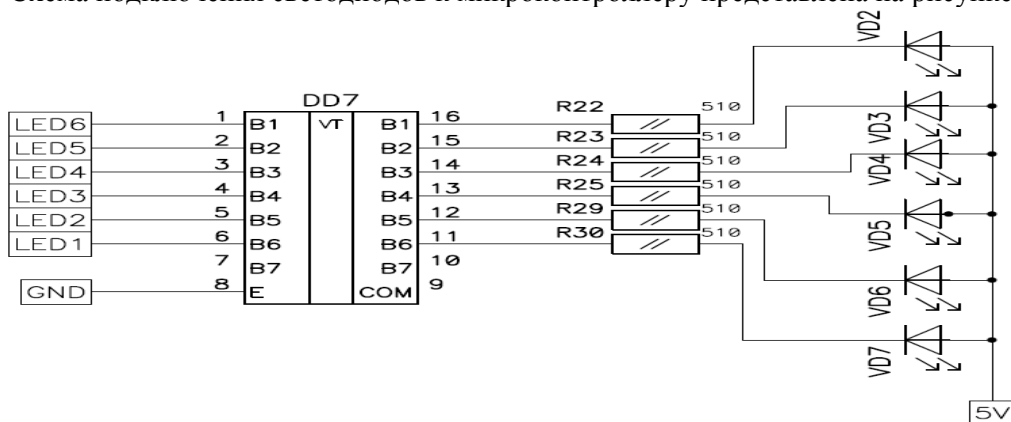


Рисунок 12 – Реализация светодиодов

Из рисунка видно, что в схеме присутствуют резисторы. Данные резисторы были добавлены в схему поскольку сопротивления, которое присутствует в микроконтроллере, может не хватить и светодиоды сгорят.

Блок питания

Питание основных компонентов платы (микроконтроллер, разъем Ethernet, RS-232 и другие) происходит от 3.3 Вольт. Питание разъема Rs-485 происходит от гальванических 5Вольт. Для получения необходимых 3.3 Вольт, используются линейные регуляторы (конверторы) (Рисунок 13).

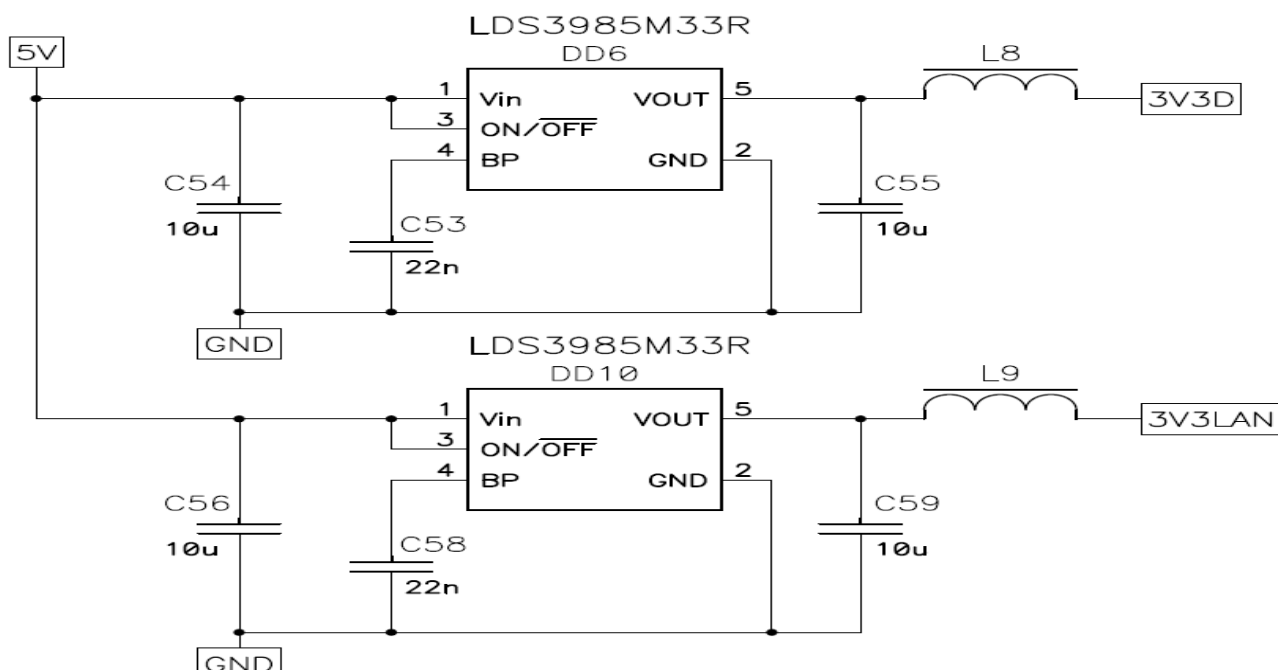


Рисунок 13 – Основное питание

На рисунке показано преобразование входного напряжения 5 Вольт в 3.3В для питания микросхем.

Применение гальванической развязки интерфейса Rs-485 обусловлено тем, что максимальный ток, протекающий между цепями, ограничивается лишь относительно небольшими электрическими сопротивлениями, что может привести к протеканию выравнивающих токов, способных причинить вред как компонентам цепи, так и людям, прикасающимся к незащищенному оборудованию. Обеспечивающий развязку прибор специально ограничивает передачу энергии от одной цепи к другой. Схема гальванического питания представлена на рисунке 14.

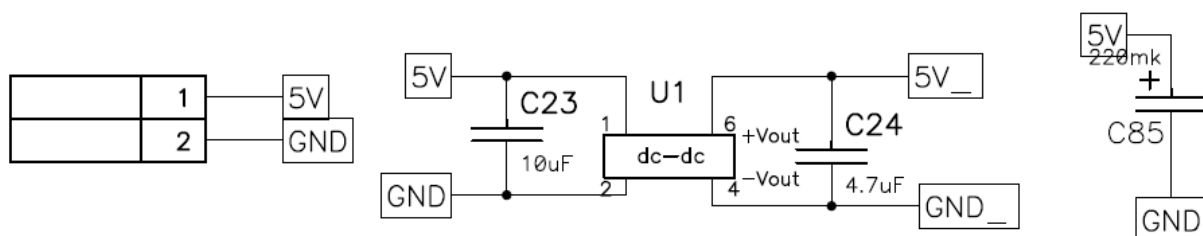


Рисунок 14 – Гальванически развязанное питание

На рисунке показано преобразование входного напряжения 5 Вольт в гальванически развязанные 5В_. Для этого применяют DC-DC преобразователь (конвертер) параллельно входам и выходам которого подключают конденсаторы.

[В начало](#)

ЧАСТЬ 2. Создание проекта в среде STM32 CubeIDE.

Разработка ПО для микроконтроллеров будет производиться при помощи бесплатной интегрированной среды разработки STM32 CubeIDE.

Для скачивания данного ПО необходимо перейти по ссылке: <https://www.st.com/en/development-tools/stm32cubeide.html>. Перейти ниже и скачать версию для своей ОС.

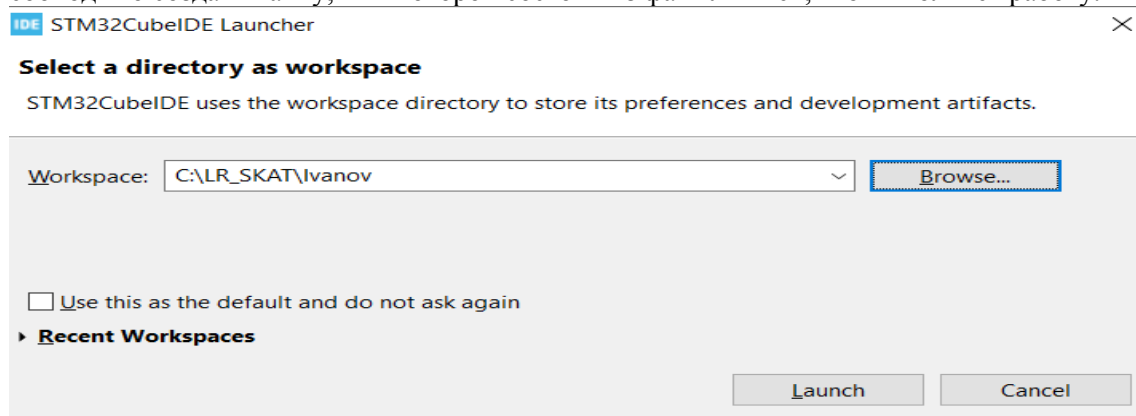
Get Software

	Part Number	General Description	Latest version	Download	All versions
+	STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.8.0	Get latest	Select version ▾
+	STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.8.0	Get latest	Select version ▾
+	STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.8.0	Get latest	Select version ▾
+	STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.8.0	Get latest	Select version ▾
+	STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.8.0	Get latest	Select version ▾

Далее производится стандартная установка.

После запуска необходимо выбрать место, где будут храниться файлы с проектами.

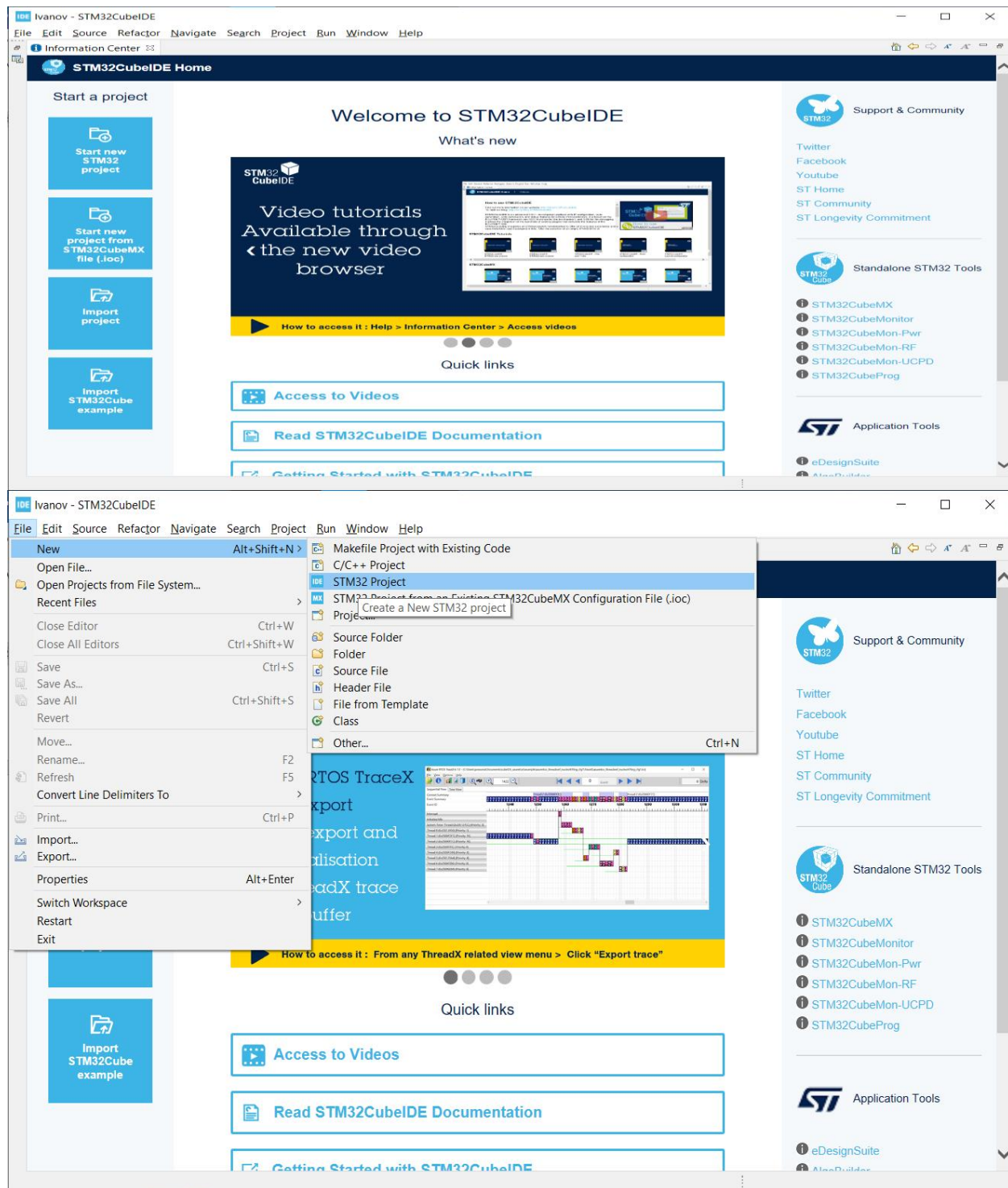
Рекомендуется выбрать папку LR_SKAT в корне диска, если она не создана, то создать её. В ней необходимо создать папку, имя которой состоит из фамилий тех, кто выполняет работу.



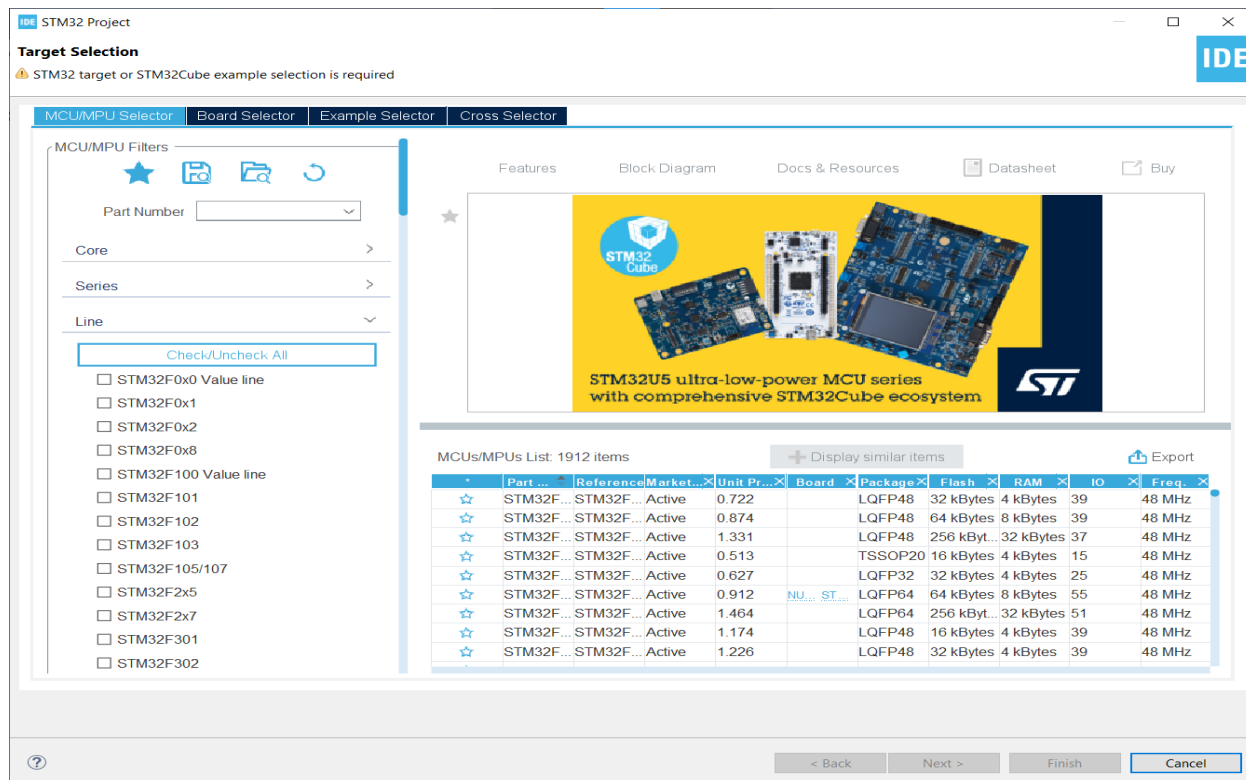
При выполнении следующих работ нужно выбирать своё рабочее пространство.

Создание проекта

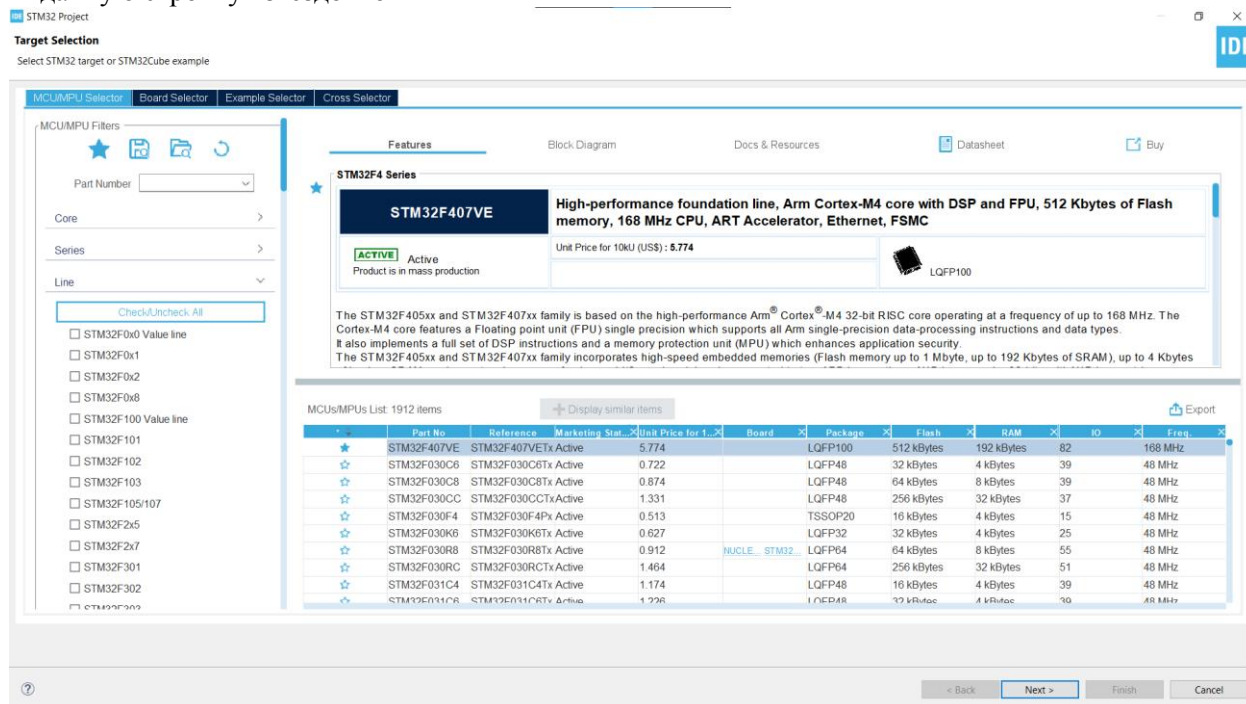
Для создания нового проекта нужно на главном экране нажать на плитку **Start new STM32 project** или **File->New-> STM32 project**



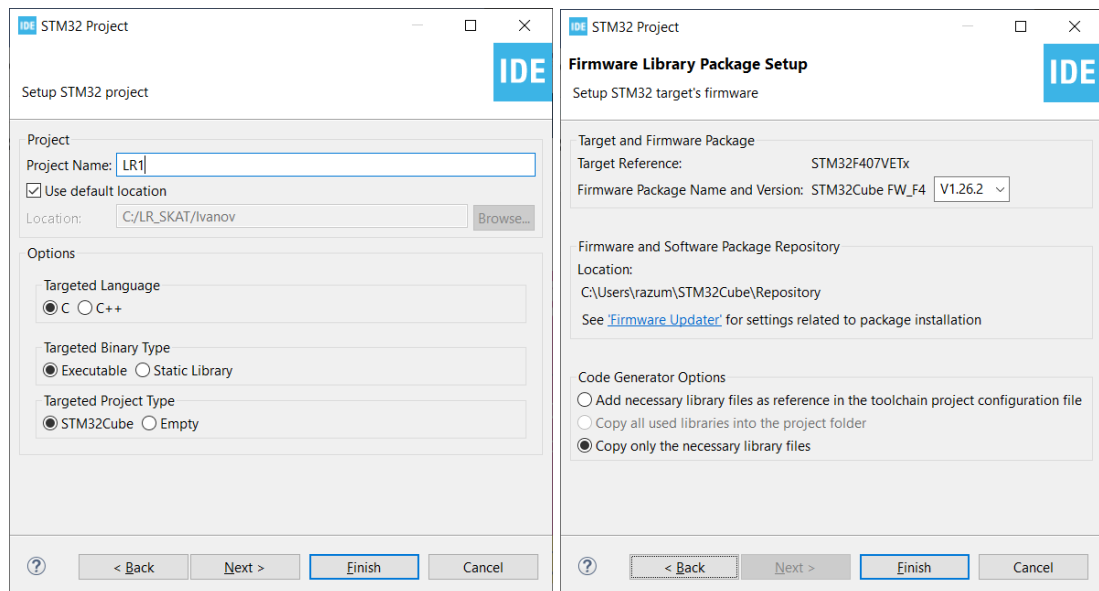
Откроется окно выбора целевого процессора



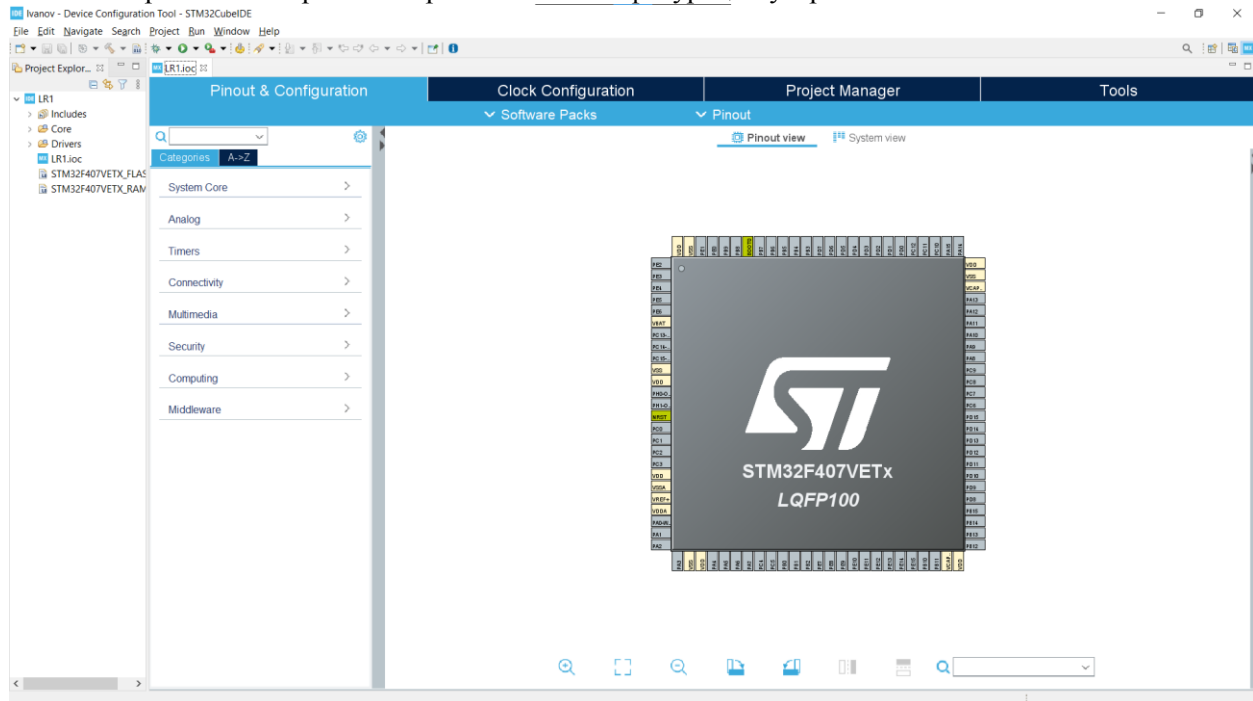
Необходимо выбрать **STM32F407VETx**, для ускорения дальнейшего поиска можно выделить данную строку «звёздочкой»



При нажатии кнопки **Next** открывается окно, в котором выбирается имя проекта и дополнительные настройки проекта.



После завершения настройки откроется окно конфигурации устройства

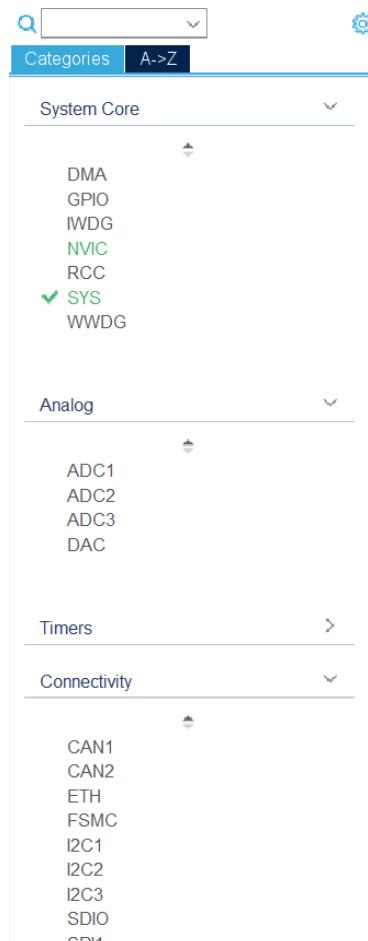


Настройка конфигурации процессора

Окно настройки конфигурации состоит из вкладок:

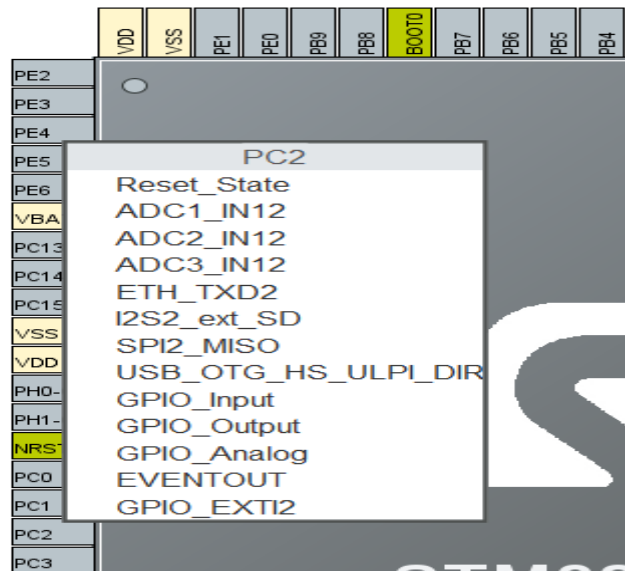
- Pinout & Configuration – содержит настройки функций выводов;
- Clock Configuration – настройки частот периферии;
- Project Manager – дополнительные настройки инициализации кода.

Вкладка **Pinout & Configuration** содержит меню, в котором представлена вся периферия процессора.

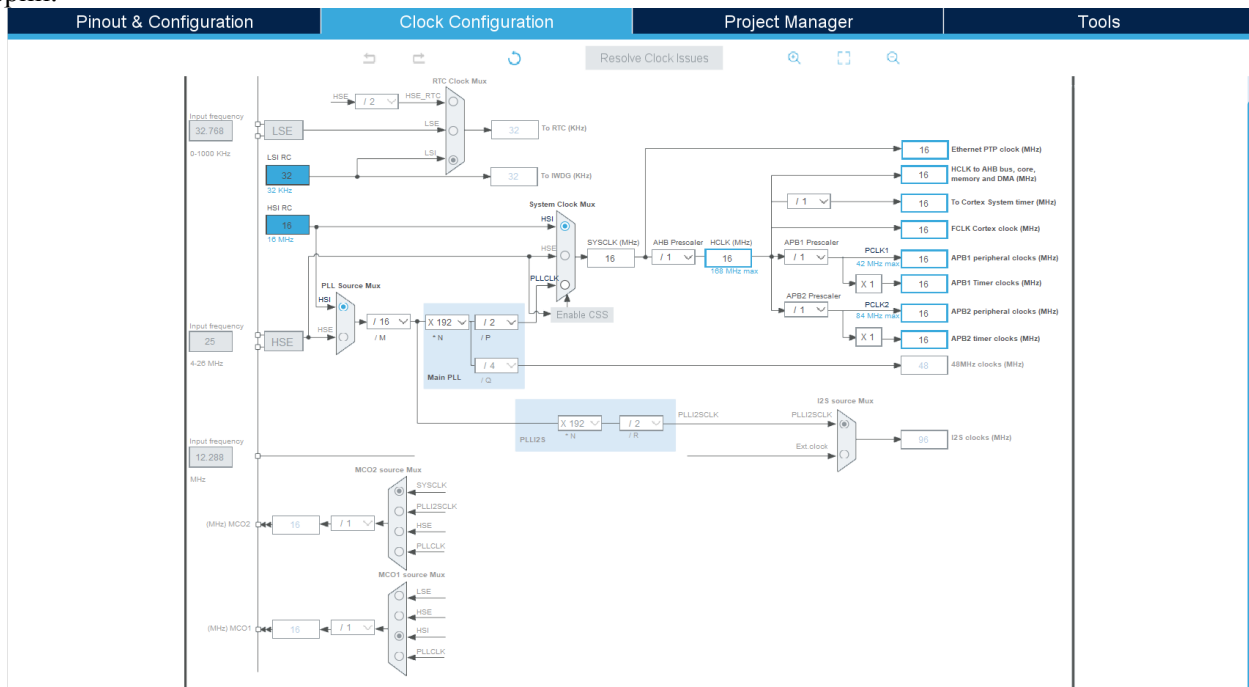


И графическую часть, в которой представлены все выводы процессора, при нажатии на любой из выводов открывается окно со всеми возможными функциями, которые можно назначить на этот вывод.

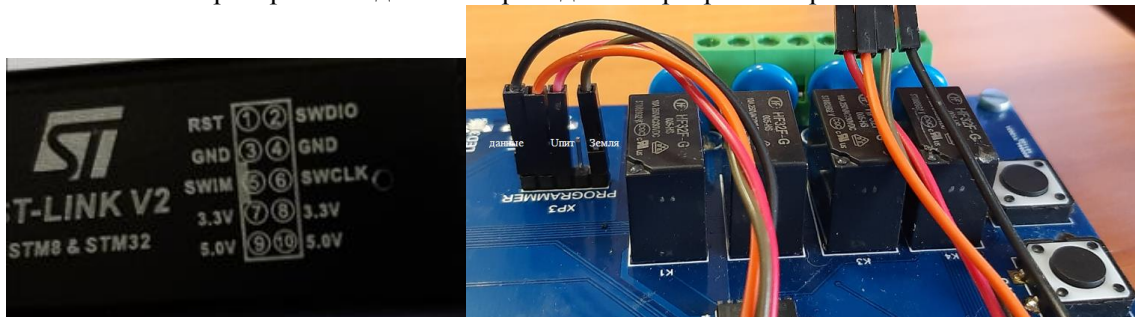




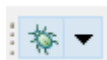
Вкладка Clock Configuration позволяет настроить частоты источников тактирования и периферии.



ВНИМАНИЕ! Проверить соединение проводов на программаторе и плате.



Для запуска программы в режиме отладки используется следующая кнопка в верхней части экрана.



Нажать кнопку и проверить работу светодиода.

Задание на лабораторную работу:

Часть 1.

1. Изучить теоретический материал части 1. Составить таблицу используемых интерфейсов.
2. Составить таблицу элементной базы для реализации 2-х блоков (питания, реле, светодиодов)
3. Описать параметры микроконтроллера по заданию преподавателя (кол-во входов, выходов, емкость памяти, flash память).
4. Выполнить загрузку и установку интегрированной среды разработки STM32 CubeIDE.
5. Создать проект в описанной выше последовательности.
6. Выполнить настройку конфигурации.
7. Создать папку для своих проектов.

Часть 2.

1. Изучить порядок разработки проекта.
2. Написать код программы, которая выполняет следующие действия каждой кнопки.
 - Первая кнопка: Нажата-первый светодиод горит, отпущена – светодиод гаснет.
 - Вторая кнопка: Нажатие меняет состояние светодиода (первое нажатие-загорается, второе-гаснет, третье загорается и т.д.)
 - Третья кнопка: По нажатию на кнопку третий светодиод мигает три раза.
 - Четвёртая кнопка: При удерживании кнопки четвёртый светодиод моргает.
 - Пятая кнопка: загораются по очереди все светодиоды.
 - Шестая кнопка: все светодиоды гаснут.

Отчет должен содержать

1. Цель работы.
2. Задание.
3. Алгоритм создания проекта
4. Листинг программы.
5. Вывод о результатах моделирования

Вопросы для самоконтроля

1. Провести сравнительный анализ характеристик микроконтроллеров stm32.
2. Указать на особенности реализации интерфейсов CAN.
3. Раскрыть особенности реализации интерфейсов CAN.
4. реализации интерфейсов CAN.
5. Привести алгоритм загрузки и установки интегрированной среды разработки STM32 CubeIDE.
6. Раскрыть особенности создания проекта.
7. Раскрыть особенности настройки тактирования.
8. Указать кол-во портов ввода-вывода микроконтроллера, их параметры.
9. Раскрыть особенности настройки портов ввода-вывода.
10. Привести алгоритм программирования портов ввода-вывода.

[В начало](#)

ЛАБОРАТОРНАЯ РАБОТА 2.

Программирование дисплея.

Цель работы: сформировать навыки программирования 4-х строчный дисплей с заданной таблицей кодировки символов.

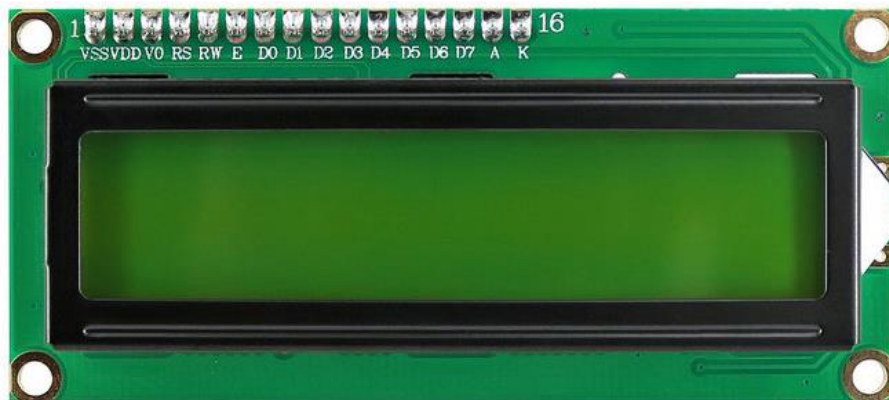
Задачи: изучить порядок программирования дисплея с заданной таблицей кодировки символов.

Порядок работы.

К данному стенду подключён 4-х строчный дисплей со следующей таблицей кодировки символов.

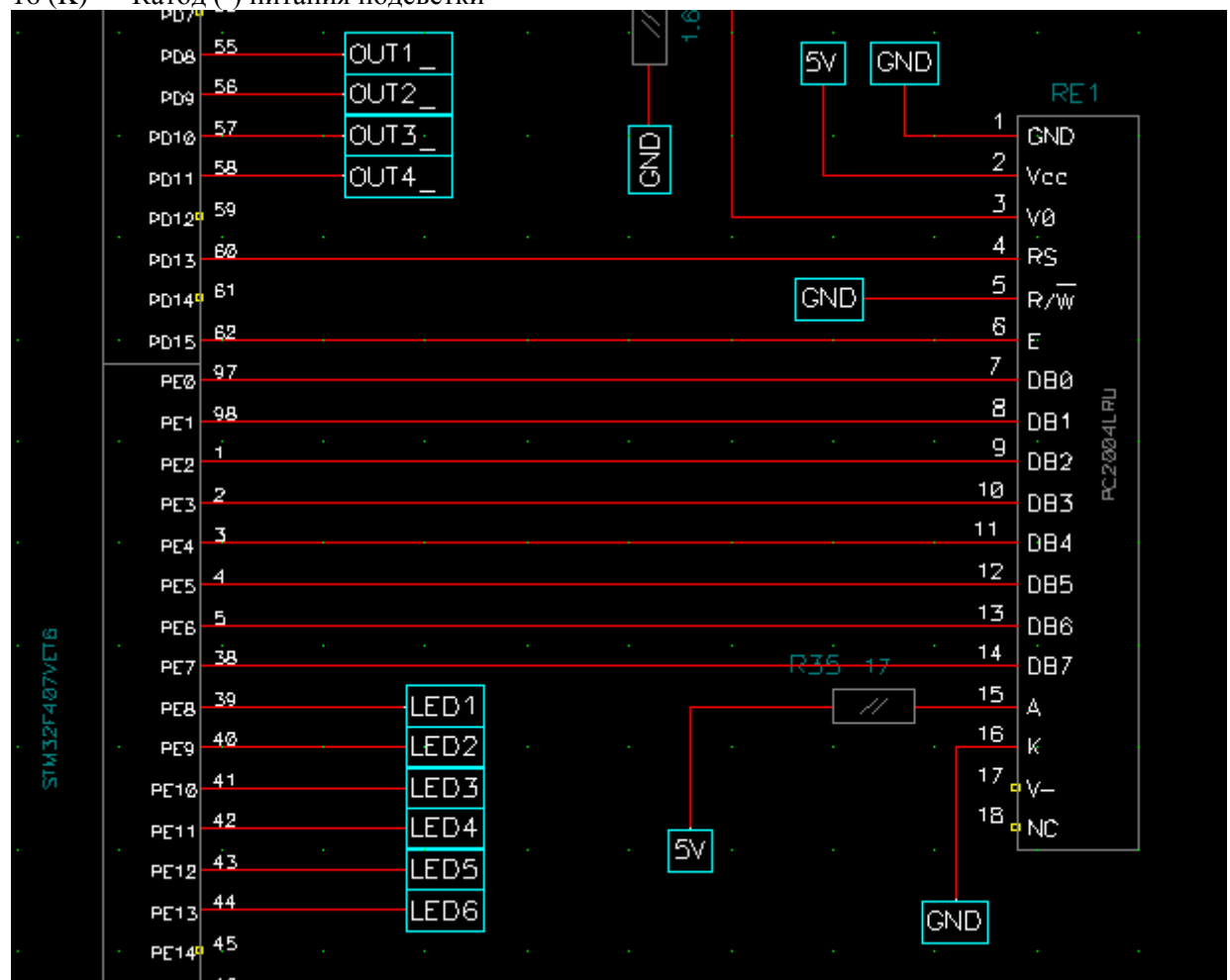
b7-b4 b3-b0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)			0	9	P	r				E	0	4	.	Д	М
0001	(2)		!	1	A	Q	a	я			Г	Я	ш	.	Ц	М
0010	(3)		"	2	B	R	b	г			ё	б	ь	и	Ш	М
0011	(4)		#	3	C	S	c	з			Ж	В	ы	"	д	Ч
0100	(5)		\$	4	D	T	d	т			Э	Г	ь	х	Ф	Н
0101	(6)		%	5	E	U	e	у			И	ё	э	х	Ц	Г
0110	(7)		&	6	F	V	f	у			Й	ж	ю	х	ш	М
0111	(8)		'	7	G	W	g	w			Л	з	я	l	'	Е
1000	(1)		(8	H	X	h	x			П	и	о	И	'	±
1001	(2))	9	I	Y	i	y			У	а	о	↑	~	±
1010	(3)		*	:	J	Z	j	z			Ф	к	а	↓	ё	±
1011	(4)		+	:	K	[k	и			Ч	л	"	к	о	±
1100	(5)		,	<	L	o	l	е			Ш	м	м	и	о	±
1101	(6)		-	=	M	I	m	е			Ъ	н	з	и	±	±
1110	(7)		.	>	N	^	n	е			Ы	п	г	х	о	±
1111	(8)		/	?	O	_	o	е			Э	т	е	.	о	■

Дисплей подключён по 8 линиям данных (DB0-DB7) .



На дисплее имеется 16-pin разъем для подключения.

- 1 (VSS) — Питание контроллера (-)
- 2 (VDD) — Питание контроллера (+)
- 3 (VO) — Вывод управления контрастом
- 4 (RS) — Выбор регистра (данные или команды)
- 5 (R/W) — Чтение/запись
- 6 (E) — Enable
- 7-10 (DB0-DB3) — Младшие биты 8-битного интерфейса
- 11-14 (DB4-DB7) — Старшие биты интерфейса
- 15 (A) — Анод (+) питания подсветки
- 16 (K) — Катод (-) питания подсветки

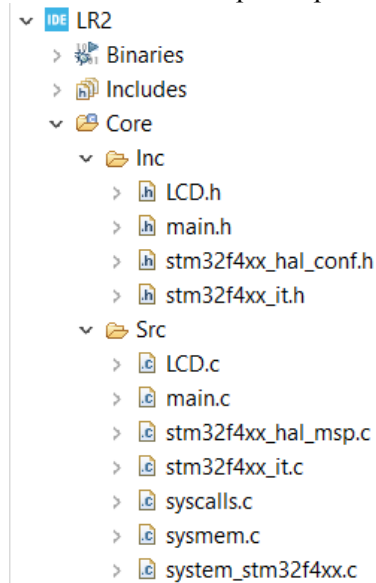


Файлы работы с дисплеем содержат в себе функции управления. Для использования этих функций необходимо добавить их в проект.

- В папку **Core/Inc** нужно скопировать заголовочный файл LCD.h

- В папку **Core/Src** нужно скопировать файл **LCD.c**

Они появятся в дереве проекта



В заголовочном файле **LCD.h** необходимо указать на каких ножках работает дисплей и прописать их в следующие определения:

```
#define d0_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_0, GPIO_PIN_SET)
#define d1_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_1, GPIO_PIN_SET)
#define d2_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_2, GPIO_PIN_SET)
#define d3_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET)
#define d4_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_4, GPIO_PIN_SET)
#define d5_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_5, GPIO_PIN_SET)
#define d6_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_6, GPIO_PIN_SET)
#define d7_set() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_7, GPIO_PIN_SET)

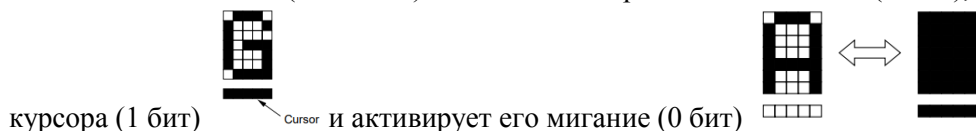
#define d0_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_0, GPIO_PIN_RESET)
#define d1_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_1, GPIO_PIN_RESET)
#define d2_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_2, GPIO_PIN_RESET)
#define d3_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET)
#define d4_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_4, GPIO_PIN_RESET)
#define d5_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_5, GPIO_PIN_RESET)
#define d6_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_6, GPIO_PIN_RESET)
#define d7_reset() HAL_GPIO_WritePin(GPIOE, GPIO_PIN_7, GPIO_PIN_RESET)

#define e1 HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET)
#define e0 HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET)
#define rs1 HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET)
#define rs0 HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET)
```

Работа с дисплеем начинается с его инициализации при помощи функции **LCD_ini()**. (Файл **LCD.c**) Этот код состоит из трёх начальных команд инициализации **LCD_WriteData(3);**.

Далее идёт команда **0x3C (00111100)**, которая активирует режим работы по 8 линиям, в 4 строки и устанавливает нужный размер каждого символа.

Команда **0x0F(00001111)** включает отображение символов (2 бит), включает индикацию



Команда 0x01 (00000001) очищает память дисплея и устанавливает курсор в нулевое положение.

Команда 0x06 (00000110) устанавливает увеличение адреса в который записывается символ после каждой записи (1 бит) и отключает сдвиг дисплея (0 бит), сдвигается только курсор.

Команда 0x02 (00000010) устанавливает адрес 0 в счетчике адресов. Также возвращает дисплей из смещенного в исходное положение. Содержимое памяти остается неизменным

Полный набор команд и их описание представлены на рисунке ниже.

Table 6 Instructions

Instruction	Code										Description	Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)	
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.		
Return home	0	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 μ s
Display on/off control	0	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 μ s
Cursor or display shift	0	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 μ s
Function set	0	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 μ s
Set CGRAM address	0	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 μ s
Set DDRAM address	0	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 μ s
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μ s
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 μ s $t_{ADD} = 4 \mu s^*$	
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 μ s $t_{ADD} = 4 \mu s^*$	
I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 \times 10 dots, F = 0: 5 \times 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable											DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{osc} is 250 kHz, $37 \mu s \times \frac{270}{250} = 40 \mu s$	

После инициализации можно использовать любые другие функции из файлов LCD(h,c).

Функция **LCD_SetPos(uint8_t x, uint8_t y);** устанавливает курсор в нужное положение на дисплее.

void LCD_String(char* st); записывает любую строку на текущее положение курсора.
 (срока- это массив символов)
void LCD_Clear(void); очищает дисплей и устанавливает курсор в нулевое положение
void LCD_SendChar(uint8_t ch); отправляет один символ на экран.
void LCD_Data(uint8_t dt); отправляет любой символ по коду из таблицы кодировки на экран

void LCD_Command(uint8_t dt); отправляет команду на дисплей
void LCD_WriteData(uint8_t dt); отправляет код на ножки (d0-d7) дисплея, используется во всех функциях


Отличие LCD_Command от LCD_Data только в том, что первая функция устанавливает ножку r/s в 1, что означает дальнейшую запись данных в регистр команд, а если r/s=0, то запись производится в регистр данных.

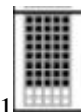


Пример вывода строки Hello World! и символа с кодом 11010010 () на дисплей.

```
/* USER CODE BEGIN 2 */
LCD_ini();
LCD_String(str1);
LCD_SetPos(0,1);
LCD_String(str2);
LCD_SendChar('!');
LCD_Data(0b11010010);
LCD_Clear();
/* USER CODE END 2 */
```

Задание на лабораторную работу:

1. Создать новый проект.
2. Настроить порты данных (PE0-PE7), RS (PD13), E (PD15) на выход, настроить тактирование процессора и настроить кнопки (JLP1)
3. Сгенерировать код
4. Подключить и настроить файлы работы с дисплеем. LCD.c и LCD.h
5. Вывести на экран строку **Hello World!** .
6. Выполнить следующее задание:
 Назначить функции кнопкам:
 Первая Кнопка: Очистка дисплея
 Вторая кнопка: Смещение курсора влево
 Третья кнопка: Смещение курсора вправо
 Четвёртая кнопка: смещение курсора вверх
 Пятая кнопка: смещение курсора вниз



Шестая кнопка: Установка символа с кодом 0b11111111

7. Вывести на экран свою фамилию.

Отчет должен содержать

1. Цель работы.
2. Задание.
3. Алгоритм создания проекта
4. Листинг программы с комментариями по лабораторной работе.
5. Листинг программы с комментариями по индивидуальному заданию.
6. Вывод о результатах моделирования

Вопросы для самоконтроля

1. Раскрыть особенности настройки кодирования дисплеев.
2. Привести алгоритм создания проекта .
3. Раскрыть порядок инициализации.

ЛАБОРАТОРНАЯ РАБОТА 3.

Работа с внешней Flash-памятью.

Цель работы: сформировать навыки программирования по интерфейсу SPI.

Задачи: разработать программу чтения и записи во внешнюю память

Порядок работы.

Для сохранения какой-либо информации применяют внешнюю память.

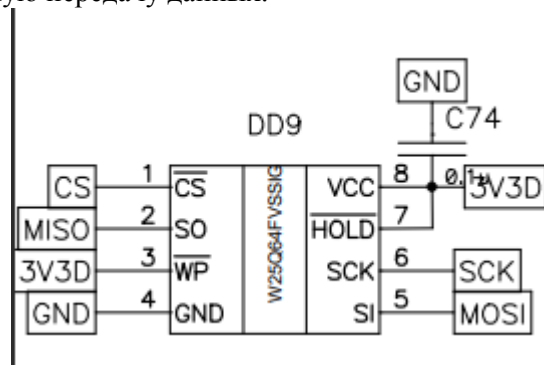
На данной плате память подключается по интерфейсу SPI.

Название интерфейс SPI является аббревиатурой от “Serial Peripheral Bus”, что можно перевести как “шина для подключения периферийных устройств”. Интерфейс использует три провода для подключения внешних устройств.

- **MOSI** - Master Output Slave Input (Ведущий передает, Ведомый принимает),
- **MISO** - Master Input Slave Output (Ведущий принимает, Ведомый передает),
- **SCLK**, иначе SCK - Serial Clock (тактовый сигнал).

Так как устройств к шине можно подключить одновременно несколько, но при этом они не обладают уникальными идентификаторами, нужен какой-то способ отличать одно от другого. Ведущий должен точно знать, кому он отправляет данные и от кого их принимает. Для этого в протокол добавлен провод SS - Slave Select.

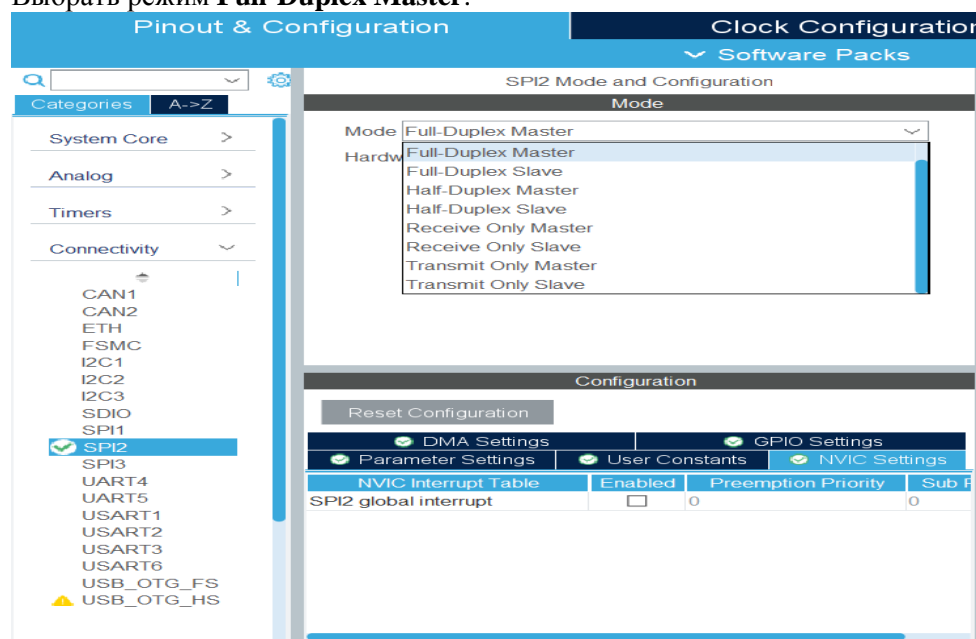
У каждого Ведомого есть для этого отдельный пин, за состоянием которого он наблюдает, падение в низкий уровень означает, что Ведущий обращается конкретно к нему и они начинают активную передачу данных.



Микросхема памяти подключается по проводам **MOSI (PC3)**, **MISO(PC2)**, **SCLK(PB10)** и **CS(PA15)**.

Настройка портов процессора для работы с SPI.

После создания нового проекта, нужно зайти на вкладку **Connectivity** и найти там **SPI2**. Выбрать режим **Full-Duplex Master**.



В графической части нужные выходы автоматически выберутся.



Далее в окне **Parameter Settings** настраиваются параметры в соответствии с рисунком, именно с такими параметрами работает внешняя память.

Mode

Mode: Full-Duplex Master

Hardware NSS Signal: Disable

Configuration

Reset Configuration

DMA Settings

GPIO Settings

Parameter Settings

User Constants

NVIC Settings

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Frame Format: Motorola

Data Size: 8 Bits

First Bit: MSB First

Clock Parameters

Prescaler (for Baud Rate): 2

Baud Rate: 8.0 MBits/s

Clock Polarity (CPOL): Low

Clock Phase (CPHA): 1 Edge

Advanced Parameters

CRC Calculation: Disabled

NSS Signal Type: Software

Сигнал выбора микросхемы (CS) автоматически не назначается, необходимо настроить выход **PA15** на выход (JP1). Также настраивается тактирование (JP1).

Генерируется код.

Для работы с внешней памятью необходимо подключить к проекту файлы ([stm32_eval_spi_flash.h](#) и [stm32_eval_spi_flash.c](#)), по аналогии с JP2.

В файле [stm32_eval_spi_flash.h](#) необходимо указать порт к которому подключена линия CS (A15) и какой используется spi (hspi2)

```
extern SPI_HandleTypeDef hspi2;
#define sFLASH_CS_GPIO_PORT    GPIOA
#define sFLASH_CS_PIN          GPIO_PIN_15
#define sFLASH_SPI              &hspi2
```

После этого можно пользоваться любыми функциями этих файлов.

void sFLASH_EraseSector(uint32_t SectorAddr);-стирает информацию из сектора по его начальному адресу
void sFLASH_EraseBulk(void); полностью очищает всю информацию
void sFLASH_WritePage(uint8_t* pBuffer, uint32_t WriteAddr, uint16_t NumByteToWrite);
 Записывает информацию из буфера в пределах одной страницы по начальному адресу и количеству байт для записи
void sFLASH_WriteBuffer(uint8_t* pBuffer, uint32_t WriteAddr, uint16_t NumByteToWrite);
 записывает информацию из буфера в несколько страниц последовательно по начальному адресу и количеству байт для записи
void sFLASH_ReadBuffer(uint8_t* pBuffer, uint32_t ReadAddr, uint16_t NumByteToRead);;- считывает информацию в буфер по начальному адресу и количеству байт для считывания.
uint32_t sFLASH_ReadID(void);;-считывает индивидуальных идентификатор чипа памяти

Вся память разделена на страницы по 256 байт, которые группируются в сектора. Можно записывать информацию, на незаполненные места, но для перезаписи нужно стирать полностью сектор

Пример кода для записи информации представлен ниже.

```
/* USER CODE BEGIN PV */
uint8_t buf[10];
uint8_t buf1[]="Hello";
uint8_t buf2[]="World";
/* USER CODE END PV */

.....

/* USER CODE BEGIN 2 */
sFLASH_EraseSector(0);           //Очистка сектора
sFLASH_WriteBuffer(buf1,0,5);    //Запись Hello (0 адрес, 5 байт)
sFLASH_ReadBuffer(buf,0,10);     //Чтение 10 байт в buf по 0 адресу (Результат:
HelloFFFF (FFFF-пустые ячейки))
sFLASH_WriteBuffer(buf2,5,5);    //Запись World (5 адрес, 5 байт)
sFLASH_ReadBuffer(buf,0,10);     //Чтение 10 байт в buf по 0 адресу (Результат:
HelloWorld)
/* USER CODE END 2 */
```

Задание на лабораторную работу

1. Создать новый проект.
2. Настроить SPI2 и порт CS(PA15) на выход
3. Сгенерировать код
4. Подключить и настроить файлы работы с внешней памятью. [stm32_eval_spi_flash.h](#) и [stm32_eval_spi_flash.c](#)
5. Выполнить запись и чтение какой-либо информации.
6. Проверить правильность выполнения в отладке.

Отчет должен содержать

1. Цель работы.
2. Задание.
3. Алгоритм создания проекта
4. Листинг программы с комментариями по лабораторной работе.
5. Листинг программы с комментариями по индивидуальному заданию.
6. Вывод о результатах моделирования

Вопросы для самоконтроля

1. Раскрыть особенности настройки портов.
2. Привести характеристики порта SPI
3. Привести алгоритм создания проекта .
4. Раскрыть порядок настройки файлов работы с внешней памятью.
5. Раскрыть характеристики внешней памяти.

ЛАБОРАТОРНАЯ РАБОТА 4.

Разработка программы обмена данными по интерфейсам.

Цель работы : сформировать навыки программирования операций обмена по различным интерфейсам.

Задачи: разработать программу функционирования кнопок, дисплея и памяти. Данная работа является частью домашнего задания.

Порядок выполнения работы:

Создать новый проект в соответствии алгоритмами рассмотренными ранее.

Задание на лабораторную работу

1. Настроить SPI2 и выход CS (PA15) на выход, настроить тактирование процессора и кнопки (JP1), настроить дисплей (JP2)
2. Сгенерировать код
3. Подключить и настроить файлы работы с внешней памятью. [stm32_eval_spi_flash.h](#) и [stm32_eval_spi_flash.c](#), файлы для работы с дисплеем [LCD.c](#) и [LCD.h](#)
4. Назначить функции кнопкам:

Первая Кнопка: Очистка внешней памяти

Вторая кнопка: Запись текста из заранее созданной переменной.

Третья кнопка: Считывание текста из внешней памяти

Процесс выполнения действий по нажатию кнопок должен комментироваться на экране.

Пример:

при нажатии первой кнопки на экране должно выводиться следующее:

Производится очистка сектора
Очистка завершена

Вторая кнопка:

Запись текста: *(текст для записи)*
Запись завершена

Третья кнопка:

Считывание данных из памяти
Результат: *(считанный текст)*

Отчет должен содержать

1. Цель работы.
2. Задание.
3. Алгоритм создания проекта
4. Листинг программы с комментариями по лабораторной работе.
5. Вывод о результатах моделирования

Вопросы для самоконтроля

1. Раскрыть особенности настройки кодирования дисплеев.
2. Привести алгоритм создания проекта .
3. Указать кол-во портов ввода-вывода микроконтроллера, их параметры.
4. Раскрыть особенности настройки портов ввода-вывода.
5. Привести алгоритм программирования портов ввода-вывод.

[В начало](#)

СПИСОК ЛИТЕРАТУРЫ

1. Олифер, В. Г. Основы сетей передачи данных : учебное пособие для СПО / В. Г. Олифер, Н. А. Олифер. — Саратов : Профобразование, 2021. — 219 с. — ISBN 978-5-4488-1007-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/102200.html> (дата обращения: 25.04.2021). — Режим доступа: для авторизир. Пользователей
2. Архитектуры и топологии многопроцессорных вычислительных систем : учебник / А. В. Богданов, В. В. Корхов, В. В. Мареев, Е. Н. Станкова. — 3-е изд. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 135 с. — ISBN 978-5-4497-0322-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/89420.html> (дата обращения: 25.04.2021). — Режим доступа: для авторизир. Пользователей
3. Шмырин, А. М. Дискретная математика и математическая логика : учебное пособие для СПО / А. М. Шмырин, И. А. Седых. — 2-е изд. — Липецк, Саратов : Липецкий государственный технический университет, Профобразование, 2020. — 160 с. — ISBN 978-5-88247-960-1, 978-5-4488-0751-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/92827.html> (дата обращения: 25.04.2021). — Режим доступа: для авторизир. пользователей. - DOI: <https://doi.org/10.23682/92827>
4. Микропроцессорные системы : учебное пособие для вузов / Е. К. Александров, Р. И. Грушвицкий, М. С. Куприянов [и др.] ; под редакцией Д. В. Пузанков. — 2-е изд. — Санкт-Петербург : Политехника, 2020. — 936 с. — ISBN 978-5-7325-1098-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/94828.html> (дата обращения: 25.04.2021). — Режим доступа: для авторизир. Пользователей
5. Алаев, П. Е. Математическая логика : учебное пособие для СПО / П. Е. Алаев, Л. Л. Максимова. — Саратов, Москва : Профобразование, Ай Пи Ар Медиа, 2020. — 98 с. — ISBN 978-5-4488-0789-3, 978-5-4497-0450-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/96015.html> (дата обращения: 25.04.2021). — Режим доступа: для авторизир. Пользователей
6. Практическое руководство по программированию STM-микроконтроллеров : учебное пособие / С. Н. Торгаев, М. В. Тригуб, И. С. Мусоров, Д. С. Чертихина. — Томск : Томский политехнический университет, 2015. — 111 с. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/55205.html> (дата обращения: 25.04.2021). — Режим доступа: для авторизир. Пользователей

[В начало](#)