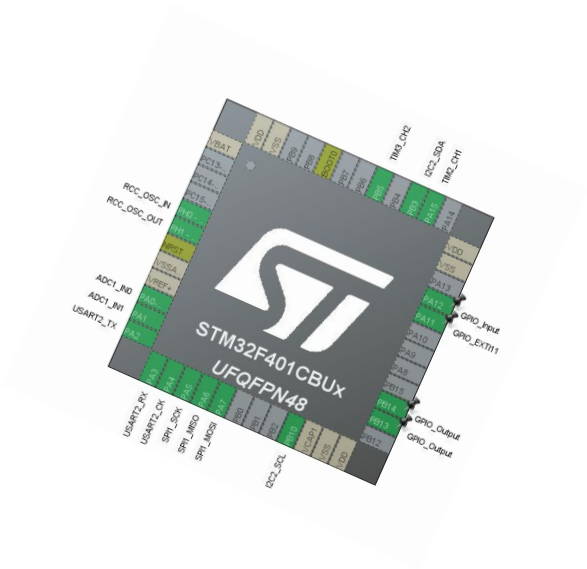
Работа с микроконтроллерами STM32
в программной среде разработки STM32CubeIDE

Методические указания
к практическим занятиям по дисциплине
«Программирование микроконтроллеров»



Практическое занятие №1

«Изучение основных принципов программирования микроконтроллеров в среде STM32CubeIDE»

Цель работы: ознакомиться со средой разработки STM32CubeIDE, сформировать проект с использованием графического интерфейса.

1 Теоретические сведения

Среда разработки STM32CubeIDE, разработанная производителем микроконтроллеров STM фирмой ST Microelectronics, позволяет проводить настройку программной и аппаратной базы под имеющийся микроконтроллер.

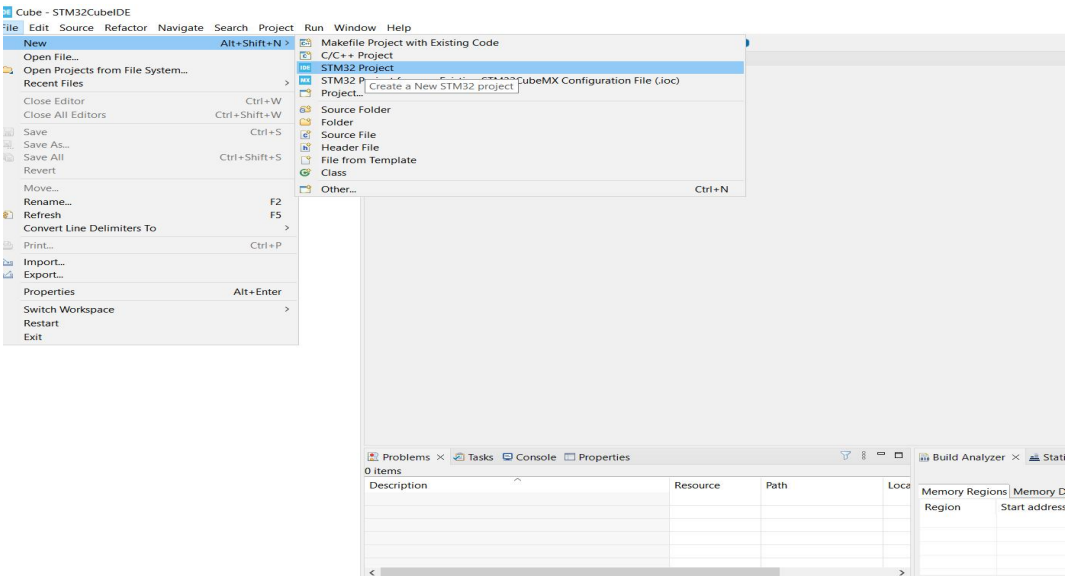
Базовые семейства STM32F0, STM32F1, STM32F3 - сбалансированные характеристики и компромиссные значения производительности, потребления и цены. STM32F1 – исторически первая линейка 32-битных микроконтроллеров от ST, построено на ядре Cortex-M3 с рабочей частотой до 72 МГц.

Высокопроизводительные семейства STM32F2, STM32F4, STM32F7, STM32H7 ориентированы на достижение максимального быстродействия.

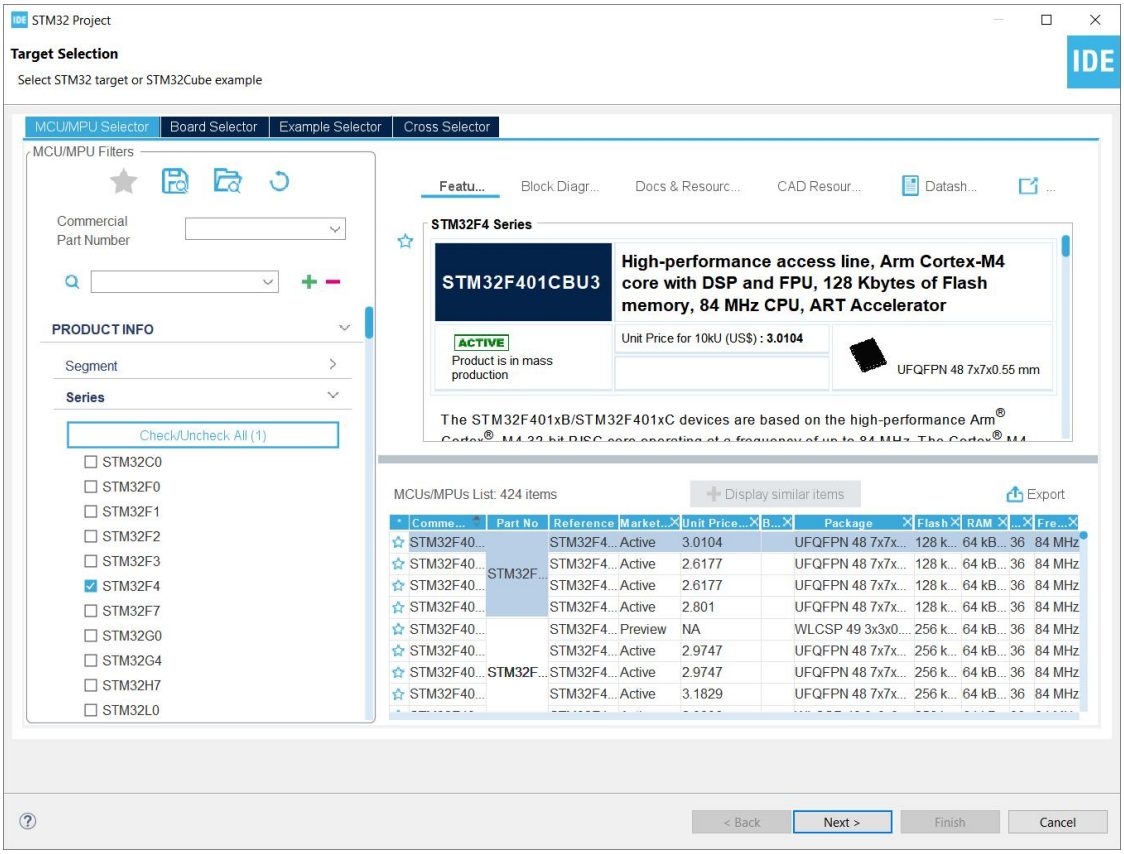
2 Порядок выполнения работы

2.1 Запустите программную среду разработки STM32CubeIDE;

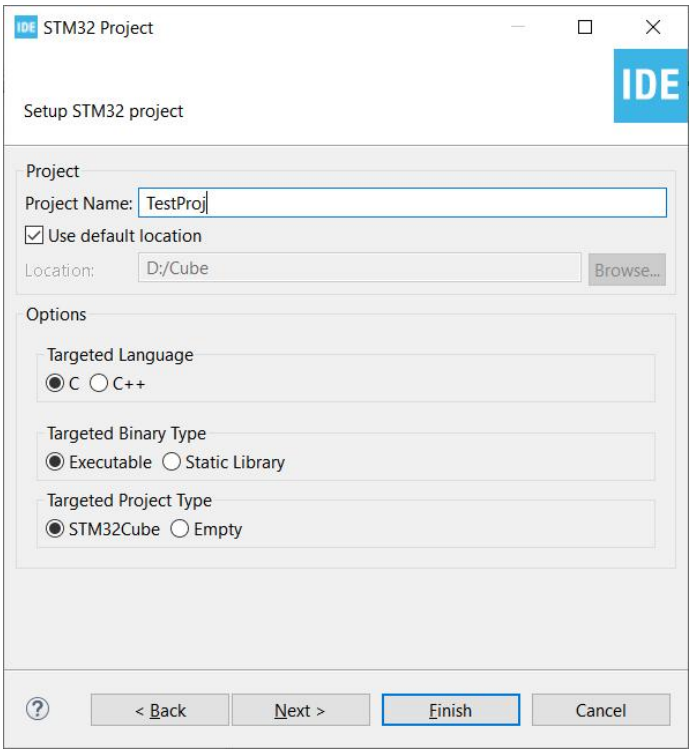
2.2 Создайте проект: *«New-STM32 Project»*



2.3 Выберите модель микроконтроллера STM32F401CBU3

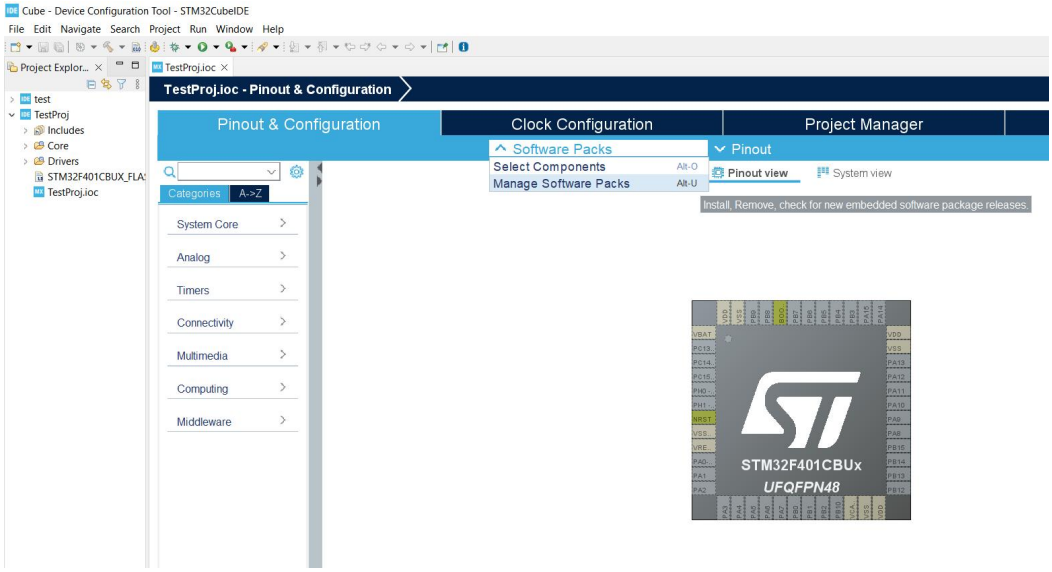


2.4 Задайте имя проекта

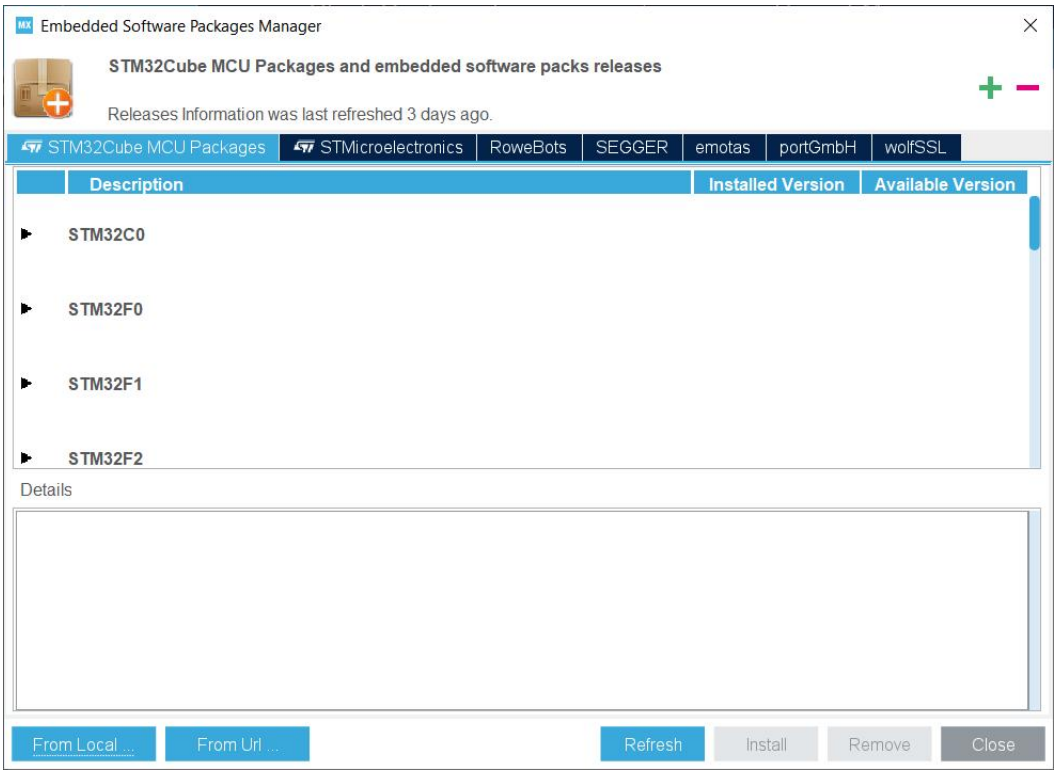


2.5 Завершите создание проекта

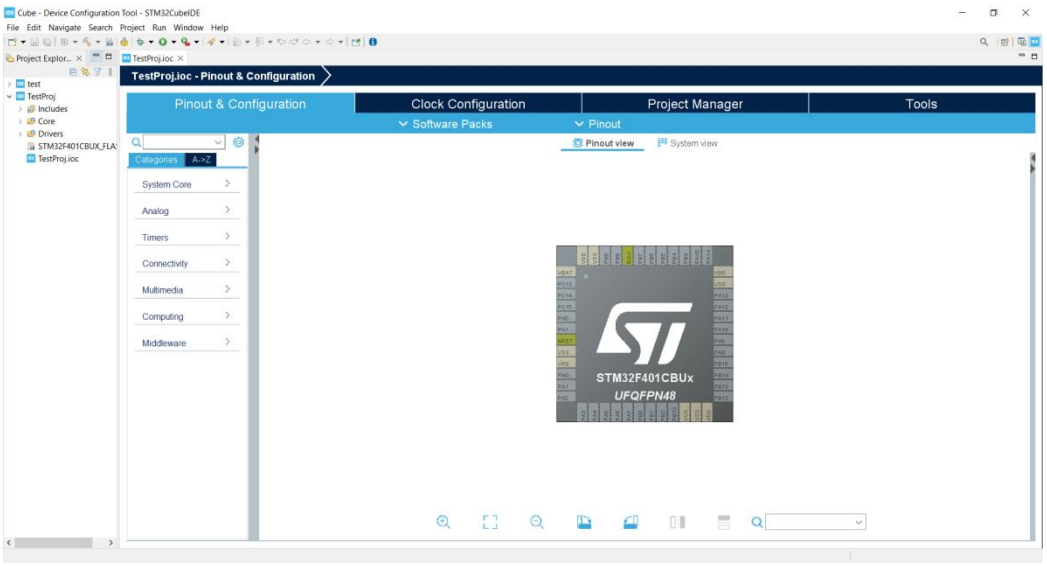
2.6 Выполните подключение программного пакета для выбранного семейства микроконтроллера: *«Software Packs - Manage Software Packs»*



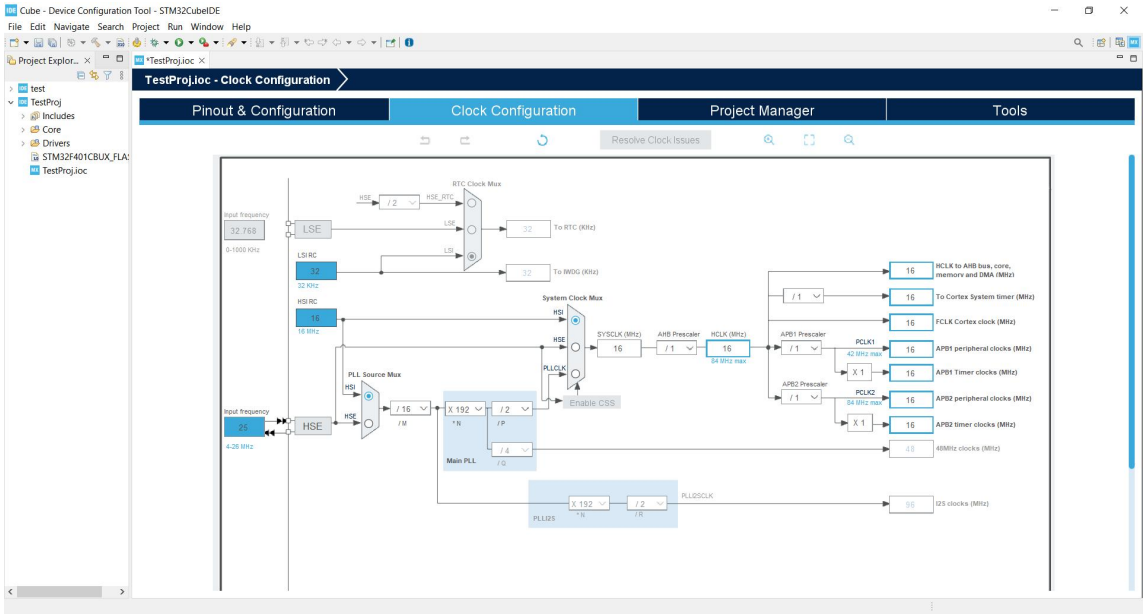
2.7 Выберите программный пакет для локальной установки: *«From Local…»* и установите пакет из папки: *«P:\bbs\STM\en.stm32cubef4_v1-27-0.zip»*. Дождитесь завершения распаковки программного пакета в папку проекта.



2.8 Выполните подключение перифифейринных устройств и настройку портов микроконтроллера в окне *«Pinout & Configuration»*



2.9 Выполните настройку блока фазовой автоподстройки частоты (PLL) и настройку частот внутренних модулей в окне *«Clock Configuration»*.



2.10 Выполните генерацию программного кода, соответствующего реализованным настройкам.

3 Содержание отчета

По результатам выполнения задания внести в отчет:

– графическое представление распределения портов микроконтроллера;

– графическое представление настройки частот внутренних модулей.

Практическое занятие №2

«Функции управления портами ввода-вывода в среде STM32CubeIDE»

Цель работы: изучить функции управления портами ввода-вывода средствами библиотеки HAL в среде разработки STM32CubeIDE.

1 Теоретические сведения

HAL — библиотека для создания приложений на STM32. Возможности среды разработки STM32CubeIDE позволяют получить автоматически сгенерированный код для периферийного окружения микроконтроллера с использованием библиотеки HAL. Наиболее простым периферийным устройством микроконтроллера являются порты ввода-вывода, которые используются для взаимодействиями с внешними подключаемыми устройствами. С их помощью осуществляется прием от внешних устройств либо передача во внешние устройства. В микроконтроллерах STM32 порты именуются буквами A, B, C и так далее: GPIOA, GPIOB, GPIOC. Каждый GPIO имеет 16 линий ввода/вывода, причем каждая линия может быть настроена независимо от других.

Контакты микросхемы, соответствующие порту ввода-вывода GPIOA обозначаются *PA#*, где # - номер вывода порта ввода-вывода, для порта GPIOB - *PB#* и т.д.

Взаимодействие с портами ввода-вывода осуществляется через регистры - особый вид памяти внутри микроконтроллера, который используется для управления процессором и периферийными устройствами, представляет собой ячейку памяти и имеет длину в 32 бита.

Регистры портов ввода-вывода:

– регистр режима - GPIOx_MODER - определяет направление работы вывода;

– регистр типа вывода - GPIOx_OTYPER - определяет тип подключения выводов, настроенных на выход;

– регистр скорости вывода - GPIOx_OSPEEDR - определяет скорость работы выводов, настроенных на выход;

– регистр режима подтягивающего резистора - GPIOx_PUPDR;

– регистр состояния выводов, настроенных на вход - GPIOx_IDR;

– регистр состояния выводов, настроенных на выход - GPIOx_ODR;

– регистр установки/сброса состояния выводов - GPIOx_BSRR;

– регистр блокировки конфигурации - GPIOx_LCKR;

– младшая часть регистра переключения портов ввода-вывода на альтернативную функцию - GPIOx_AFRL;

– старшая часть регистра переключения портов ввода-вывода на альтернативную функцию - GPIOx_AFRH.

Подробное описание регистров портов ввода-вывода приведено в справочном руководство по микроконтроллеру STM32F401CB - RM0368 Reference manual.

2 Порядок выполнения работы

2.1 Выполните настройку портов ввода-вывода в графическом интерфейсе среды разработки STM32CubeIDE в соответствии с вариантом, приведенным в таблице 1.

Таблица 1 - Варианты заданий практического занятия №1

Номер варианта	Аналоговый вход	Вход	Выход
1	*PA5*	*PC14* *Pull-up*	*PA6* *Output push pull* *Pull-up*
2	*PA11*	*PA1* *Pull-up*	*PB1* *Output push pull* *Pull-down*
3	*PA8*	*PA4* *Pull-down*	*PC13* *Output open drain* *Pull-up*
4	*PA6*	*PB3* *Pull-up*	*PB2* *Output open drain* *Pull-up*
5	*PB1*	*PC13* *Pull-up*	*PA12* *Output push pull* *Pull-down*
6	*PB10*	*PA10* *Pull-down*	*PC15* *Output push pull* *Pull-up*
7	*PA7*	*PB13* *Pull-down*	*PA10* *Output push pull* *Pull-down*
8	*PC13*	*PB7* *Pull-up*	*PA1* *Output push pull* *Pull-up*
9	*PA15*	*PC15* *Pull-down*	*PA6* *Output open drain* *Pull-up*
10	*PC15*	*PB1* *Pull-up*	*PA5* *Pull-up*

2.2 Сгенерируйте программный код, соответствующий заданной конфигурации.

2.3 Проанализируйте программный код библиотеки HAL, приведенный в файле stm32f4xx_hal_gpio.c.

2.4 Определите назначение, с указанием возможных значений, полей структур:

– программной модель порта ввода-вывода;

– инициализации порта ввода-вывода.

2.5 Сформируйте описание работы функций:

– инициализации порта ввода-вывода;

– деинициализации вывода;

– чтения состояния вывода;

– установки состояния вывода;

– инвертирования состояния вывода;

– блокировки конфигурации.

2.6 Выделите и опишите программный код, реализующий программную настройку в соответствии с вариантом задания:

– включение тактирования портов ввода-вывода;

– инициализацию портов ввода-вывода.

3 Пример описания программного кода

В качестве примера описания программных функций рассматривается функция установки состояния порта ввода-вывода:

Функция установки состояния порта ввода-вывода - *HAL_GPIO_WritePin*, расположена в файле *stm32f4xx_hal_gpio.c*.

Возвращаемое значение функции: *void* - нет возвращаемого значения.

Параметры функции:

1) GPIOx - указатель на структуру программной модели порта ввода-вывода - определяет используемый порт ввода-вывода;

2) GPIO_Pin - целочисленное беззнаковое 16-битное число - определяет номер вывода порта ввода-вывода;

3) *PinState* - значение перечисляемого типа - определяет устанавливаемое состояние вывода.

Краткий алгоритм работы функции:

1) выполняется проверка передаваемых в функцию параметров

assert_param(IS_GPIO_PIN(GPIO_Pin));

assert_param(IS_GPIO_PIN_ACTION(PinState));

2) для значений состояния вывода не равных сброшенному состоянию, выполняется установка бита состояния в регистре сброса/установки значения вывода

```
if(PinState != GPIO_PIN_RESET)
{
  GPIOx->BSRR = GPIO_Pin;
}
```

3) для значений состояния вывода равных сброшенному состоянию, выполняется сброс бита состояния в регистре сброса/установки значения вывода

```
else
{
  GPIOx->BSRR = (uint32_t)GPIO_Pin << 16U;
}
```

4 Содержание отчета

По результатам выполнения задания внести в отчет описание программных функций и структур данных библиотеки HAL, используемых для управления портами ввода-вывода.

Практическое занятие №3

«Функции управления прерываниями в среде STM32CubeIDE»

Цель работы: изучить функции управления прерываниями от внешних линий средствами библиотеки HAL в среде разработки STM32CubeIDE.

1 Теоретические сведения

Управление аппаратными средствами связано с асинхронными событиями. Большинство из них приходит от аппаратной периферии. Все микроконтроллеры предоставляют функцию, называемую прерываниями. Прерывание – это асинхронное событие, которое вызывает остановку выполнения текущего кода в приоритетном порядке. Прерывания являются источником многозадачности: аппаратное обеспечение знает о них и отвечает за сохранение текущего контекста исполнения перед вызовом обработчика прерывания. Они используются операционными системами реального времени для введения понятия задач. Без помощи аппаратного обеспечения невозможно создать настоящую вытесняющую многозадачность, которая позволяет переключаться между несколькими контекстами исполнения без необратимой потери текущего исполняемого потока. Прерывания могут возникать как от аппаратного, так и от программного обеспечения. Архитектура ARM различает два типа исключений: прерывания, вызываемые аппаратным обеспечением, и исключения, вызываемые программным обеспечением. Процессоры Cortex-M предоставляют модуль, предназначенный для управления исключениями. Он называется Контроллером вложенных векторных прерываний (Nested Vectored Interrupt Controller, NVIC).

Когда микроконтроллер STM32 загружается, по умолчанию разрешены только исключения Reset, NMI и Hard Fault. Остальные исключения и периферийные прерывания запрещены, и они должны быть разрешены при возникновении запроса IRQ. Чтобы разрешить IRQ, CubeHAL предоставляет следующую функцию:

void HAL_NVIC_EnableIRQ(IRQn_Type IRQn);

соответствующая функция для запрета IRQ:

void HAL_NVIC_DisableIRQ(IRQn_Type IRQn);

где *IRQn_Type* – это перечисление всех исключений и прерываний, определенных для конкретного микроконтроллера. Перечисление IRQn_Type является частью HAL от ST и определяется в заголовочном файле модели микроконтроллера.

Микроконтроллеры STM32 предоставляют различное количество источников внешних прерываний, подключенных к контроллеру NVIC через контроллер EXTI,

который, в свою очередь, способен управлять несколькими линиями запроса внешних прерываний контроллера EXTI. Соотношение приведено на рисунке 1.

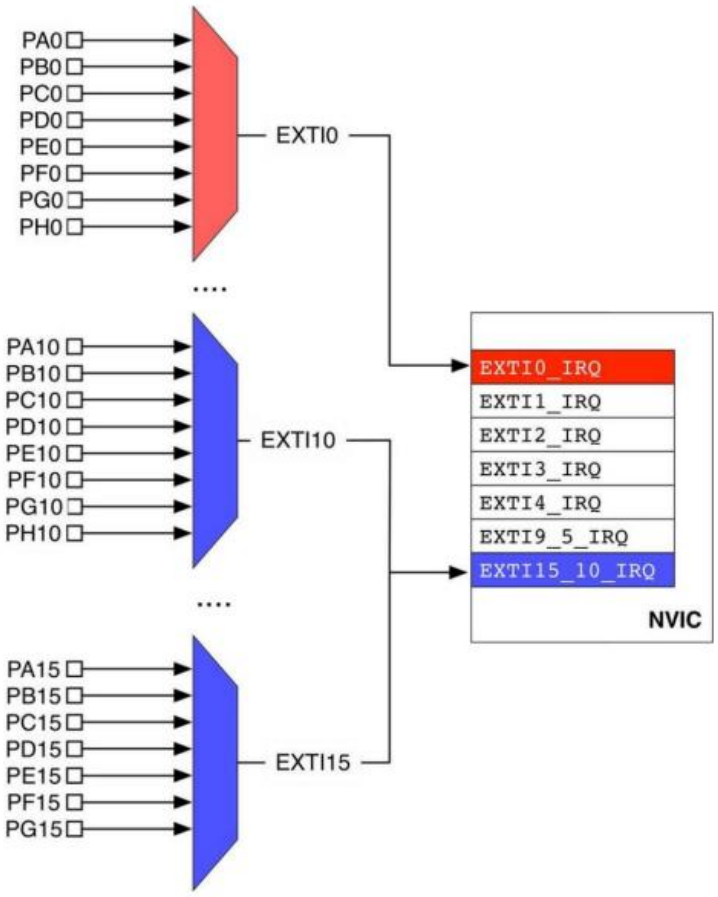


Рисунок 1 - Соотношение между линиями GPIO и линиями запроса прерываний EXTI в микроконтроллерах STMF4

STM32CubeIDE можно использовать для облегчения разрешения IRQ и автоматической генерации кода ISR. Первым шагом является разрешение соответствующей линии запроса прерывания EXTI, используя представление Chip. Как только мы разрешили IRQ, нам нужно дать CubeMX команду генерировать соответствующую ISR. Данная конфигурация выполняется в представлении Configuration, нажав кнопку NVIC. Появится список ISR, которые можно разрешить.

2 Порядок выполнения работы

2.1 Выполните настройку внешних линий прерываний в графическом интерфейсе среды разработки STM32CubeIDE в соответствии с вариантом, приведенным в таблице 1.

Таблица 1 - Варианты заданий практического занятия №3

Номер варианта	Внешняя линия прерывания
1	*PA0*
2	*PA1*
3	*PA5*
4	*PA11*
5	*PB0*
6	*PB10*
7	*PB12*
8	*PC13*
9	*PB3*
10	*PC14*

2.2 Сгенерируйте программный код, соответствующий заданной конфигурации.

2.3 Проанализируйте программный код библиотеки HAL, приведенный в файлах:

– startup_stm32f401cbux.s

– stm32f4xx_it.c

– stm32f4xx_hal_exti.c

2.4 Сформируйте описание программного кода:

– доступных обработчиков прерываний;

– функции запрета всех прерываний;

– функции разрешения всех прерываний;

– функции установки приоритета прерывания;

– фунции разрешения прерывания по номеру;

– фунции запрета прерывания по номеру.

3 Содержание отчета

По результатам выполнения задания внести в отчет:

– конфигурацию линии внешнего прерывания;

– описание программных функций и структур данных библиотеки HAL, используемых для управления прерываниями.

Практическое занятие №4

«Функции управления таймерами в среде STM32CubeIDE»

Цель работы: изучить функции управления таймерами средствами библиотеки HAL в среде разработки STM32CubeIDE.

1 Теоретические сведения

Таймеры имеют 4 независимых канала, которые могут использоваться для:

– захвата сигнала (Timer Input Capture) - позволяет измерять период следования импульсов: при приходе импульса запоминается значение счетного регистра, при приходе следующего импульса, из текущего значения счетного регистра вычитается зафиксированное ранее значение;

– сравнения (Output Compare) - при достижении счетным регистром определнного значения значение вывода изменяется в соответствии с настройками (0, 1 или изменение на противоположное);

– генерации широтно-импульсной модуляции (PWM Generation);

– генерации одиночного импульса (Forced Output).

Таймеры 16-битные, умеют работать с инкрементальными энкодерами и датчиками Холла, несколько таймеров можно синхронизировать между собой. Возможность настройки прерываний по событиям:

– переполнение;

– захват сигнала;

– сравнение;

– событие-триггер.

Регистры таймера:

– TIMx_CNT - регистр счетчика, прямое обращение к регистру счетчика необходимо только в случае отслеживания времени без использования прерывания, циклическими проверками значения таймера;

– TIMx_PSC - регистр предделителя;

– TIMx_ARR - регистр перезагрузки;

– TIMx_RCR - регистр счетчика повторов;

– TIMx_CR1 - регистр управления, задает режим работы таймера, структура приведена на рис. 1

Рис.1 - Структура регистра управления таймера

– биты 9-8 (CKD) - задают значение делителя частоты для внутренних нужд таймера;

– бит 7 (ARPE) определяет режим записи в регистр перезагрузки;

– биты 6-5 (CMS) определяют режим для двунаправленного счета (01-11) или разрешают счет только в одном направлении (00);

– бит 4 (DIR) задает направление счета: прямой или реверсивный (0 или 1);

– битом 3 (OPM) можно задать режим одиночного импульса. Если он равен 1, то при перезагрузке счетчик останавливается;

– бит 2 (URS) меняет логику формирования события UEV. При 0, к естественному формированию события перезагрузки таймера добавляются, программная инициация события с помощью бита UG и перезагрузка от ведомого таймера;

– бит 1 (UDIS) управляет разрешением перезагрузки счетчика. Если задать его равным 1, то автоматическая перезагрузка счетчика происходить не будет;

– бит 0 (CEN) разрешает работу счетчика. Если он не установлен, то счетчик полностью отключен;

– TIMx_DIER - регистр разрешения прерываний: бит 0 (UIE) – разрешение прерывания по перезагрузке. Прерывание разрешено при состоянии бита равном 1.

– TIMx_SR - регистр состояния, содержит флаги прерываний: бит 0 (UIF) – флаг прерывания по переполнению.

– TIMx_EGR - регистр генерации событий, позволяет искусственно формировать события: 0 (UG) –генерация перезагрузки. Производится записью в него 1. Сбрасывается аппаратно;

– TIMx_SMCR - регистр управления ведомым режимом таймера: младшие 3 бита (SMS) - для того чтобы задать тактирование таймера от шины APB, надо записать в них нули.

Настройка таймеров в среде разработки STM32CubeIDE:

Для настройки таймера выберите его в окне «Timers» конфигурирования системы. Активируйте таймер, задав поле «Clock Source». Выполните конфигурацию таймера в окне «Configuration», обязательно установите параметры «Prescaler» (указываемое

значение должно быть на 1 меньше требуемого) и «Counter Period». Включите прерывание таймера в окне настроек «NVIC Settings».

2 Порядок выполнения работы

2.1 Выполните настройку таймеров в графическом интерфейсе среды разработки STM32CubeIDE в соответствии с вариантом, приведенным в таблице 1. Период работы таймеров и направление счета выберите самостоятельно.

Таблица 1 - Варианты заданий практического занятия №4

Номер варианта	Таймер в режиме захвата сигнала	Таймер в режиме сравнения	Таймер в режиме ШИМ	Предделитель
1	*TIM1*	*TIM3*	*TIM2*	*4*
2	*TIM2*	*TIM1*	*TIM4*	*128*
3	*TIM3*	*TIM4*	*TIM2*	*1024*
4	*TIM4*	*TIM2*	*TIM1*	*512*
5	*TIM2*	*TIM3*	*TIM4*	*2*
6	*TIM4*	*TIM1*	*TIM2*	*2048*
7	*TIM3*	*TIM2*	*TIM1*	*256*
8	*TIM1*	*TIM3*	*TIM2*	*16*
9	*TIM4*	*TIM2*	*TIM3*	*32*
10	*TIM2*	*TIM1*	*TIM3*	*8*

2.2 Сгенерируйте программный код, соответствующий заданной конфигурации.

2.3 Проанализируйте программный код библиотеки HAL, приведенный в файлах:

– stm32f4xx_it.c - обработчики прерываний таймера;

– stm32f4xx_hal_tim.c - функции работы с таймерами.

2.4 Проанализируйте настройки таймеров в файле main.c.

2.5 Сформируйте описание программного кода:

– обработчиков прерываний таймеров;

– функций инициализации таймеров;

– функций работы с таймерами.

3 Содержание отчета

По результатам выполнения задания внести в отчет:

– параметры настройки таймеров;

– графическое представление конфигурации;

– описание программных функций и структур данных библиотеки HAL, используемых для управления таймерами.

Практическое занятие №5

«Работа с последовательными интерфейсами в режиме DMA в среде STM32CubeIDE»

Цель работы: изучить средства конфигурирования последовательных интерфейсов и контроллера DMA в среде разработки STM32CubeIDE.

1 Теоретические сведения

В настоящее время в электронной промышленности существует очень большое количество протоколов последовательной связи и аппаратных интерфейсов. Некоторые из этих стандартов пришли из прошлого, но все еще широко распространены, особенно в качестве интерфейса связи между модулями на плате. Одним из них является интерфейс универсального синхронно-асинхронного приемопередатчика (Universal Synchronous/Asynchronous Receiver/Transmitter), также известного как USART.

UART/USART – это устройство, передающее параллельную последовательность битов (обычно сгруппированных в байтах) в непрерывном потоке сигналов, протекающих по одному проводу. Почти каждый микроконтроллер имеет как минимум одно периферийное устройство UART. Почти все микроконтроллеры STM32 предоставляют по крайней мере два интерфейса UART/USART, но большинство из них предоставляют более двух интерфейсов

Для конфигурации периферийного устройства UART/USART необходимо разрешение периферийного устройства в представлении Pinout. Используемые выводы будут автоматически выделены зеленым цветом. Затем в разделе Configuration необходимо сконфигурировать параметры USART, такие как скорость передачи, длину слова и так далее. После конфигурации интерфейса USART можно сгенерировать код Си. STM32CubeIDE помещает весь код инициализации USART в функцию MX_USART_Init(), который содержится в файле main.c, а весь код, связанный с конфигурацией GPIO, помещается в функцию HAL_UART_MspInit(), которая находится в файле stm32xxxx_hal_msp.c.

Двумя наиболее распространенными протоколами для внутриплатного обмена данными являются I^2C и SPI.

Двухпроводный синхронный протокол I^2C предназначен для средне скоростного обмена данными в локальной сети (до 128 устройств) между микроконтроллерами или между микроконтроллером и периферийными устройствами.

В STM32CubeIDE для включения периферийного устройства I^2C необходимо воспользоваться представлением Pinout, после чего можно сконфигурировать параметры

в представлении Configuration. Используемые выводы будут автоматически выделены зеленым цветом.

Последовательный периферийный интерфейс (Serial Peripheral Interface, SPI) – это спецификация последовательной, синхронной и полнодуплексной связи между контроллером ведущего устройства (обычно микроконтроллером) и несколькими ведомыми устройствами. Задание конфигурации интерфейса SPI в среде STM32CubeIDE аналогично рассмотренным последовательным интерфейсам.

Все рассмотренные последовательные интерфейсы могут работать в режиме приема и передачи с использованием контроллера DMA.

Контроллер прямого доступа к памяти (Direct Memory Access, DMA) – это специализированный и программируемый аппаратный модуль, позволяющий периферийным устройствам микроконтроллера получать доступ к внутренней памяти без вмешательства ядра.

Микроконтроллеры семейств STM32F2/F4/F7 имеют два контроллера DMA. В микроконтроллерах семейств STM32F2/F4/F7 каждый контроллер DMA реализует 8 разных потоков. Каждый поток предназначен для управления запросами на доступ к памяти с одного или нескольких периферийных устройств. Каждый поток может иметь до 8 каналов (запросов) и арбитра для обработки приоритета между запросами к DMA.

Кроме того, каждый поток может по выбору предоставлять (является вариантом конфигурации) 32-разрядный буфер FIFO глубиной в четыре слова. FIFO используется для временного хранения данных, поступающих из источника, до его передачи в пункт назначения.

STM32CubeIDE позволяет снизить до минимума усилия, необходимые для конфигурации запросов канала/потока. После включения периферийного устройства в разделе Pinout, необходимо перейти в раздел Configuration и нажать кнопку DMA. Появится диалоговое окно DMA Configuration. Диалоговое окно содержит две вкладки, которые относятся к периферийным запросам. Например, чтобы разрешить запрос к DMA для USART2 в режиме передачи типа память-в-периферию, необходимо нажать кнопку Add и выбрать запись USART2_TX. STM32CubeIDE автоматически заполнит оставшиеся поля. Затем возможно назначение приоритета запросу и установка других параметров, таких как режим работы DMA, инкрементирование адресов периферийных устройств/памяти и т. д. Таким же образом можно сконфигурировать каналы/потоки DMA для передач типа память-в-память.

2 Порядок выполнения работы

2.1 Выполнить настройку последовательного интерфейса обмена данными в соответствии с вариантом в графическом интерфейсе среды разработки STM32CubeIDE. Варианты приведены в таблице 1.

Таблица 1 - Варианты заданий практического занятия №5

Номер варианта	Последовательный интерфейс	Режим работы
1	*I2C1*	*I2C*
2	*I2C2*	*I2C*
3	*I2C3*	*I2C*
4	*SPI1*	*Full-Duplex Master*
5	*SPI2*	*Full-Duplex Master*
6	*SPI3*	*Full-Duplex Master*
7	*USART1*	*Synchronous/CTS/RTS*
8	*USART1*	*Asynchronous*
9	*USART2*	*Synchronous/CTS/RTS*
10	*USART2*	*Asynchronous*

2.2 Выполнить подключение приемника и передатчика последовательного интерфейса обмена данными в настройках контроллера DMA (номер контроллера DMA определяется типом подключаемого интерфейса).

2.3 Выполнить подключение прерываний контроллера последовательного интерфейса и контроллера DMA в настройках контроллера прерываний NVIC.

2.4 Сгенерировать программный код, соответствующий заданной конфигурации.

2.5 Проанализировать функции настройки контроллеров последовательного интерфейса и DMA - файл main.c.

2.6 Проанализировать программный код библиотеки HAL, приведенный в файлах:

– stm32f4xx_hal_dma_ex.c;

– stm32f4xx_hal_dma.c;

– stm32f4xx_it.c;

– stm32f4xx_hal_msp.c

– stm32f4xx_hal_uart.c/stm32f4xx_hal_spi.c/stm32f4xx_hal_i2c.c - согласно варианту задания.

2.7 Сформировать описание проанализированного программного кода.

3 Содержание отчета

По результатам выполнения задания внести в отчет:

– графическое представление конфигурации последовательного интерфейса;

– графическое представление конфигурации DMA;

– графическое представление распределение портов ввода-вывода;

– описание программного кода инициализации используемых периферийных устройств;

– описание программных функций и структур данных библиотеки HAL, используемых для управления используемым последовательным интерфейсом и DMA.

Практическое занятие №6

«Функции управления аналого-цифровым преобразованием в среде STM32CubeIDE»

Цель работы: изучить функции управления аналого-цифровым преобразованием библиотеки HAL в среде разработки STM32CubeIDE.

1 Теоретические сведения

Все микроконтроллеры STM32 имеют как минимум один аналого-цифровой преобразователь (АЦП) – периферийное устройство, используемое для получения входных напряжений через специально предназначенные выводы микроконтроллера и преобразования их в числовое представление. Входные напряжения сравниваются с известным фиксированным напряжением, называемым опорным напряжением. Почти во всех микроконтроллерах STM32 АЦП реализован как 12-разрядный АЦП, последовательного приближения. Алгоритм последовательного приближения вычисляет напряжение входного сигнала путем сравнения его с генерируемым напряжением от внутреннего цифро-аналогового преобразователя, которое представляет собой часть опорного напряжения: если входной сигнал больше внутреннего опорного напряжения, то происходит дальнейшее увеличение этого опорного напряжения пока входной сигнал не станет меньше него. Окончательный результат будет соответствовать числу в диапазоне от нуля до максимального 12-разрядного целого беззнакового числа.

Рассматриваемый микроконтроллер STM32F401CBU3 имеет 10 каналов, позволяющих измерять сигналы от внешних источников, а также внутренние каналы:

– канал для внутреннего датчика температуры (VSENSE),

– канал для внутреннего опорного напряжения (VREF INT),

– канал для контроля внешнего напряжения источника питания (VBAT).

АЦП в микроконтроллерах STM32 могут преобразовывать сигналы от различного количества каналов, при этом вводится понятие группы - последовательности преобразований, которые могут быть выполнены любыми каналами и в любой очередности. Входные каналы могут быть логически переупорядочены для формирования пользовательских последовательностей выборки. Изменение порядка каналов осуществляется путем присвоения им ранга в диапазоне от 1 до 16.

Для контроллера АЦП существуют две группы каналов:

– регулярная группа (regular group), включающая в себя до 16 каналов, которая соответствует последовательности периодически сканируемых каналов во время преобразования;

– инжектированная группа (injected group), включающая в себя до 4 каналов, которая соответствует последовательности инжектированных каналов, если выполняется преобразование инжектированных каналов.

Группа инжектированных каналов имеет более высокий приоритет над группой регулярных каналов.

АЦП, реализованные в микроконтроллерах STM32, предоставляют несколько режимов преобразования:

– режим одноканального однократного преобразования - АЦП выполняет одну выборку одного канала и останавливается после завершения преобразования;

– режим многоканального однократного преобразования - АЦП выполняет одно измерение уровней сигнала на нескольких каналах в последовательности, задаваемой рангами;

– режим одноканального непрерывного преобразования - АЦП непрерывно преобразует один канал в фоновом режиме, возможно использование с DMA в циклическом режиме;

– режим многоканального непрерывного преобразования - АЦП выполняет непрерывное измерение уровней сигнала на нескольких каналах в последовательности, задаваемой рангами, осуществляется в режиме DMA;

– режим преобразования инжектированных каналов - предназначен для использования при запуске преобразования по внешнему событию или при программном запуске, прерывает преобразование текущего канала в группе регулярных каналов.

Для управления устройством АЦП используется структура библиотеки HAL ADC_HandleTypeDef. Для конфигурирования устройства АЦП - структура ADC_InitTypeDef.

CubeMX позволяет сконфигурировать периферийное устройство АЦП в несколько шагов:

– выбрать контроллер АЦП в представлении IP Tree pane;

– выбрать нужные каналы АЦП в представлении Configuration;

– сконфигурировать периферийное устройство АЦП в представлении Configuration, вкладка Parameter Settings: поле Scan Conversion Mode - определяет выполнение операции АЦП в режиме сканирования каналов, поле Continuous Conversion Mode - определяет выполнение операции АЦП в режиме непрерывного преобразования, поле End Of Conversion Selection - определяет момент установки флага конца преобразований - после каждого преобразования либо по завершению преобразования всей группы сигналов;

– сконфигурировать режим работы с регулярными каналами: установить необходимое количество преобразований, задать ранги преобразований, установить количество циклов на выполнение преобразования.

После выполнения конфигурации АЦП, выполняется конфигурация контроллера DMA - вкладка DMA Settings - необходимо добавить поток от АЦП в контроллер DMA, выполнить настройку режима работы. Далее, необходимо выполнить включение прерывания АЦП - вкладка NVIC Settings. Настройка портов ввода-вывода - вкладка GPIO Settings - выполняется автоматически.

2 Порядок выполнения работы

2.1 Выполнить настройку АЦП в графическом интерфейсе среды разработки STM32CubeIDE в соответствии с вариантом, приведенным в таблице 1. АЦП сконфигурировать в режиме многоканального многократного преборазования. В работе АЦП использовать регулярную группу каналов. Количество преобразований в одном цикле работы установить равным количеству обрабатываемых каналов. Все каналы должны быть задействованы в преобразовании.

Таблица 1 - Варианты заданий практического занятия №6

Номер варианта	Каналы АЦП
1	*AIN0, AIN4, VSENSE*
2	*AIN1, AIN3, VREF INT*
3	*AIN2, AIN3, VREF INT*
4	*AIN4, AIN9, VREF INT*
5	*AIN6,VSENSE, VREF INT*
6	*AIN5, VREF INT, VSENSE*
7	*AIN2, AIN9, VSENSE*
8	*AIN0, AIN7,VREF INT*
9	*AIN3, AIN8, VSENSE*
10	*AIN0, AIN2, VSENSE*

2.4 Сгенерировать программный код, соответствующий заданной конфигурации.

2.5 Проанализировать функции настройки контроллеров АЦП и DMA - файл main.c.

2.6 Проанализировать программный код библиотеки HAL, приведенный в файлах:

– stm32f4xx_hal_dma.c;

– stm32f4xx_it.c;

– stm32f4xx_hal_msp.c;

– stm32f4xx_hal_adc.c.

2.7 Сформировать описание проанализированного программного кода.

3 Содержание отчета

По результатам выполнения задания внести в отчет:

– параметры настройки АЦП: содержимое вкладок Parameter Settings (настройки АЦП и настройки каналов), DMA Settings, NVIC Settings, GPIO Settings;

– графическое представление конфигурации;

– описание программного кода инициализации АЦП;

– описание программных функций и структур данных библиотеки HAL, используемых для работы с АЦП.