Лабораторная работа № 3

Контроллеры ARM: внутренние прерывания

Цель работы: изучение принципиального устройства и основ программирования контроллера вложенных векторных прерываний.

1. Контроллер вложенных векторных прерываний (NVIC)

Микроконтроллер STM32F072RBT содержит разные функциональные модули (ADC, DAC, COMP, Timers, USART и другие), которые в процессе работы генерируют **аппаратные события**. Для маркировки событий используют **флаги** – биты статусных регистров. Например, событие: «приемник USART1 поместил битовый пакет в буферный регистр RDR», - сопровождается установкой флага RXNE (переключение к уровню логической «1»). Этот флаг представлен 5-ым битом статусного регистра USART1_ISR. Его периодический опрос в программе позволяет узнать о получении новой порции данных по интерфейсу USART1. При чтении данных из буферного регистра USART1_RDR флаг RXNE автоматически сбрасывается (переключение к уровню логического «0»), после чего приемник снова готов сигнализировать о получения следующего битового пакета посредством этого флага. Таким образом, возникновение аппаратных событий в системе не приводит к нарушению работы основной программы, и их обработка синхронизована с процессом ее выполнения. Недостатками подобного подхода являются задержка реакции на событие и усложнение структуры основной программы из-за необходимости слежения за множеством флагов. Минимизировать время реакции на событие позволяет **аппаратное прерывание**.

Сигнал аппаратного прерывания генерируется функциональным модулем микроконтроллера как следствие аппаратного события, если в настройках модуля присутствует соответствующее разрешение. Появление прерывания в системе приводит к приостановке основной программы и передаче управления микропроцессором функции-обработчику прерывания. По завершению обработки прерывания управление возвращается основной программе и ее выполнение продолжается с места остановки.

Микроконтроллер STM32F072RBT работает с двумя видами прерываний: **внутренними** и **внешними**. Внутренние (глобальные) прерывания генерируются в системе как отклик на события во внутренних модулях микроконтроллера, а внешние (дополнительные) прерывания поступают через порты GPIO от устройств, подключенных к выводам STM32F072RBT. Управление внутренними прерываниями выполняет **контроллер вложенных векторных прерываний (NVIC)** (nested vectored interrupt controller), а с внешними прерываниями работает **контроллер расширенных прерываний и событий (EXTI)** (extended interrupts and events controller) (рис. 1).

NVIC является элементом ядра Cortex-M0. Он контролирует сигналы аппаратных прерываний от периферийных (внутренних) модулей, в том числе от EXTI, организует приостановку и восстановление работы основной программы,

сообщает микропроцессору адрес функции-обработчика прерывания (вектор прерывания).

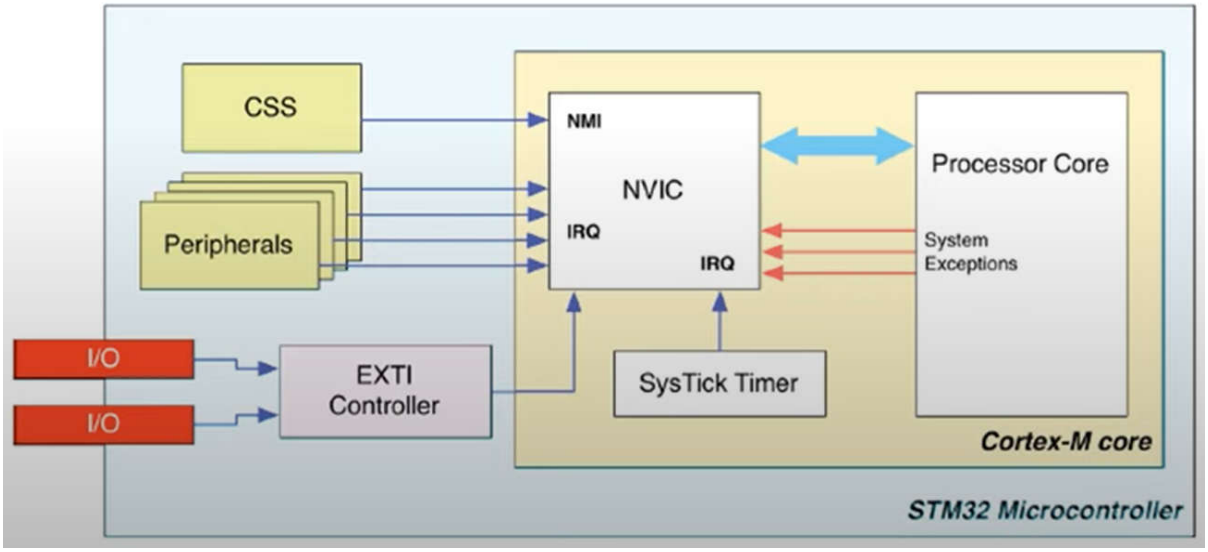

Рис. 1. Организация обработки прерываний в микроконтроллере STM32

Основные характеристики NVIC:

- 32 маскируемых канала прерывания (не считая шестнадцати линий прерываний ядра Cortex-M0);
- 4 программируемых уровня приоритета (используются 2 бита приоритета прерывания);
- Обработка исключений и прерываний с малой задержкой;
- Контроль управления питанием;
- Внедрение системных контрольных (статусных) регистров.

Процессорное ядро Cortex-M0 генерирует особые прерывания, которые называются **системными исключениями.** Сигналы исключений используют отдельные линии для подключения к NVIC и имеют более высокие уровни приоритета, чем внутренние и внешние прерывания микроконтроллера. Любые прерывания, поддерживаемые NVIC, записываются в таблицу векторов прерываний. По своей сути, таблица векторов прерываний есть ни что иное, как список адресов функций обработчиков прерываний. Номер в списке соответствует номеру прерывания (рис. 2). Каждому прерыванию при настройке NVIC присваивается свой приоритет. Если во время обработки низкоприоритетного прерывания возникает высокоприоритетное, то оно, в свою очередь, прервет обработчик низкоприоритетного прерывания.

В процессе инициации прерывания NVIC переключает ядро в режим обработки прерывания. При переходе в режим обработки прерывания регистры ядра помещаются в стек и осуществляется выборка начального адреса функции-обработчика прерывания. По завершении обработки прерывания все действия выполнятся в обратном порядке: извлекается содержимое стека и, параллельно с этим, осуществляется выборка адреса возврата.

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	All class of fault	0x0000 000C
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000 0040
1	8	settable	PVD_VDDIO2	PVD and V_{DDIO2} supply comparator interrupt (combined EXTI lines 16 and 31)	0x0000 0044
2	9	settable	RTC	RTC interrupts (combined EXTI lines 17, 19 and 20)	0x0000 0048
3	10	settable	FLASH	Flash global interrupt	0x0000 004C
4	11	settable	RCC_CRS	RCC and CRS global interrupts	0x0000 0050
5	12	settable	EXTI0_1	EXTI Line[1:0] interrupts	0x0000 0054
6	13	settable	EXTI2_3	EXTI Line[3:2] interrupts	0x0000 0058
7	14	settable	EXTI4_15	EXTI Line[15:4] interrupts	0x0000 005C
8	15	settable	TSC	Touch sensing interrupt	0x0000 0060
9	16	settable	DMA_CH1	DMA channel 1 interrupt	0x0000 0064
10	17	settable	DMA_CH2_3	DMA channel 2 and 3 interrupts	0x0000 0068
11	18	settable	DMA_CH4_5_6_7	DMA channel 4, 5, 6 and 7 interrupts	0x0000 006C
12	19	settable	ADC_COMP	ADC and COMP interrupts (ADC interrupt combined with EXTI lines 21 and 22)	0x0000 0070
13	20	settable	TIM1_BRK_UP_ TRG_COM	TIM1 break, update, trigger and commutation interrupt	0x0000 0074
14	21	settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 0078
15	22	settable	TIM2	TIM2 global interrupt	0x0000 007C
16	23	settable	TIM3	TIM3 global interrupt	0x0000 0080
17	24	settable	TIM6_DAC	TIM6 global interrupt and DAC underrun interrupt	0x0000 0084
18	25	settable	TIM7	TIM7 global interrupt	0x0000 0088
19	26	settable	TIM14	TIM14 global interrupt	0x0000 008C
20	27	settable	TIM15	TIM15 global interrupt	0x0000 0090
21	28	settable	TIM16	TIM16 global interrupt	0x0000 0094
22	29	settable	TIM17	TIM17 global interrupt	0x0000 0098
23	30	settable	I2C1	I^2C1 global interrupt (combined with EXTI line 23)	0x0000 009C
24	31	settable	I2C2	I^2C2 global interrupt	0x0000 00A0
25	32	settable	SPI1	SPI1 global interrupt	0x0000 00A4
26	33	settable	SPI2	SPI2 global interrupt	0x0000 00A8
27	34	settable	USART1	USART1 global interrupt (combined with EXTI line 25)	0x0000 00AC
28	35	settable	USART2	USART2 global interrupt (combined with EXTI line 26)	0x0000 00B0
29	36	settable	USART3_4	USART3 and USART4 global interrupts	0x0000 00B4
30	37	settable	CEC_CAN	CEC and CAN global interrupts (combined with EXTI line 27)	0x0000 00B8
31	38	settable	USB	USB global interrupt (combined with EXTI line 18)	0x0000 00BC

Рис. 2. Таблица векторов прерывания STM32072RBT
(серым цветом помечены строки, соответствующие системным исключениям)

Регистры NVIC

Регистр ISER (программный доступ: **NVIC->ISER[0]**) используется для разрешения прерываний и просмотра статуса каналов прерываний (рис. 3).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETENA[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Рис. 3. Структура регистра ISER

При записи в регистр (биты от 0 до 31 соответствуют каналам прерываний): 0 – без эффекта; 1 – разрешение прерывания. При чтении из регистра (биты от 0 до 31 соответствуют каналам прерываний): 0 – прерывание запрещено; 1 - прерывание разрешено. Примечание. Если отложенное прерывание изменяет состояние на разрешенное, то NVIC активирует его в зависимости от приоритета. Если прерывание запрещено, то при появлении сигнала изменяется состояние прерывания на ожидающее, но NVIC не активирует прерывание независимо от его приоритета.

Регистр ICER (программный доступ: **NVIC->ICER[0]**) используется для запрещения прерываний и просмотра статуса каналов прерываний (рис. 4).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRENA[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Рис. 4. Структура регистра ICER

При записи в регистр (биты от 0 до 31 соответствуют каналам прерываний): 0 – без эффекта; 1 – запрещение прерывания. При чтении из регистра (биты от 0 до 31 соответствуют каналам прерываний): 0 – прерывание запрещено; 1 - прерывание разрешено.

Регистр ISPR (программный доступ: **NVIC->ISPR[0]**) используется для переключения прерывания в состояние ожидания и показывает, какие прерывания ожидают обработки (рис. 5).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETPEND[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETPEND[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Рис. 5. Структура регистра ISPR

При записи в регистр (биты от 0 до 31 соответствуют каналам прерываний): 0 – без эффекта; 1 – изменение состояния прерывании на ожидающее. При чтении из

регистра (биты от 0 до 31 соответствуют каналам прерываний): 0 – прерывание не является ожидающим; 1 – прерывание является ожидающим. Примечание. Запись 1 в бит ISPR, соответствующий ожидающему прерыванию, не оказывает воздействие на состояние прерывания. Запись 1 в бит ISPR, соответствующий отключенному прерыванию, изменяет состояние этого прерывания на ожидающее.

Регистр ICPR (программный доступ: **NVIC->ICPR[0]**) используется для сброса состояния ожидания прерываний и показывает, прерывания ожидают обработки (рис. 6).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRPEND[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRPEND[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Рис. 6. Структура регистра ICPR

При записи в регистр (биты от 0 до 31 соответствуют каналам прерываний): 0 – без эффекта; 1 – удаляет состояние ожидания у прерывания. При чтении из регистра (биты от 0 до 31 соответствуют каналам прерываний): 0 – прерывание не является ожидающим; 1 – прерывание является ожидающим. Примечание. Запись 1 в бит ICPR не влияет на активное состояние соответствующего прерывания.

Регистры IPRx (программный доступ: **NVIC->IP[x]**) (x – число от 0 до 7) используется для установки приоритета группы прерываний. Для каждого прерывания регистры IPR предоставляют 8-битное поле приоритета. Каждый регистр содержит четыре поля приоритета, как показано на рисунке 7.

	31 … 24	23 … 16	15 … 8	7 … 0
IPR7	PRI_31	PRI_30	PRI_29	PRI_28
⋮				
IPR*n*	PRI_(4n+3)	PRI_(4n+2)	PRI_(4n+1)	PRI_(4n)
⋮				
IPR0	PRI_3	PRI_2	PRI_1	PRI_0

Рис. 7. Структура регистров IPRx

Для определения номера **x** регистра IPRx следует разделить нацело номер прерывания на 4. При этом остаток от целочисленного деления покажет смещении байта (от 0 до 3) внутри 32-битного слова IPRx. Например, прерывание USART1 имеет в микроконтроллере STM32F072RBT номер 27. Результатом целочисленного деления на 4 будет номер x = 6, а остаток, равный 1, укажет смещение байта с данными о приоритете 27 прерывания. В итоге, параметры приоритета прерывания USART1 задаются в 8-15 битах регистра IPR6.

Процессор микроконтроллера использует только биты [7:6] каждого поля приоритета прерываний. Биты [5:0] всегда читаются как 0, а при записи значения в них игнорируются. Таким образом, прерывания с 0 по 31 можно отнести к 4 группам уровней приоритета. Чем меньше значение в поле приоритета прерывания, тем выше приоритет прерывания. Например, при задании уровня приоритета прерывания USART1 7 бит поля приоритета соответствует 15 биту регистра IPR6, а 0 бит поля приоритета соответствует 8 биту регистра IPR6. Чтобы отнести прерывание USART к группе с наименьшим приоритетом, следует в биты [15:14] регистра IPR6 поместить единичные значения.

Глобальное управление прерываниями

Ассемблер STM32072RTB содержит инструкции **CPSIE I** и **CPSID I**, которые глобально разрешают и запрещают прерывания в системе. CMSIS реализует выполнение этих инструкций с помощью функций языка C++:

```
void __disable_irq(void);          //Глобальное запрещение прерываний в системе
void __enable_irq(void);           //Глобальное разрешение прерываний в системе
```

Применение этих функций в коде способствует реализации атомарных операций, которые не могут быть прерваны обработчиками прерываний.

Конфигурирование и использование прерываний в программе

При построении программных средств с использованием прерываний нужно выполнить следующие шаги конфигурирования в программе микроконтроллера:

- *Реализовать в программе обработчик прерываний*. Для микроконтроллера STM32F072RBT названия обработчиков прерываний в CMSIS представлены в файле «startup_stm32f072xb.s» (рис. 8). Например, обработчиком прерываний от модуля USART1 является функция, имеющая следующий прототип:

  ```
  void USART1_IRQHandler(void);
  ```

  По завершении передачи или приёма кадра модуль USART1 будет сигнализировать NVIC о потребности в обработке события с помощью прерывания. Для этого NVIC инициирует переход ядра в режим обработки прерывания и предоставит из таблицы векторов прерываний адрес функции-обработчика USART1_IRQHandler. Модуль USART1 для разных событий генерирует единственный сигнал прерывания, и чтобы определить какое событие вызвало прерывание, нужно воспользоваться флагами. Если прерывание произошло по приёму будет выставлен флаг USART_ISR_RXNE, по завершении передачи — USART_ISR_TC. В процессе обработки прерывания флаг USART_ISR_TC нужно сбрасывать вручную, используя регистр ICR. Пример шаблона функции-обработчика представлен ниже:

  ```
  void USART1_IRQHandler(void)
  {
    //Прерывание произошло по завершении приёма битового пакета
    if (USART1->ISR & USART_ISR_RXNE)
    {
  ```

```
        Data=(uint8_t)USART1->RDR;   //Копирование принятых данных
                                     // в программную переменную
        //Код дальнейшей обработки события
        …
    }

     //Прерывание произошло по завершении передачи битового пакета
    if (USART1->ISR & USART_ISR_TC)
    {
        // Сброс флага завершения передачи
        USART1->ICR = USART_ICR_TC;

        // Код дальнейшей обработки события
        …
    }
}
```

```
__Vectors       DCD     __initial_sp               ; Top of Stack
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     SVC_Handler                ; SVCall Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     PendSV_Handler             ; PendSV Handler
                DCD     SysTick_Handler            ; SysTick Handler

                ; External Interrupts
                DCD     WWDG_IRQHandler            ; Window Watchdog
                DCD     PVD_VDDIO2_IRQHandler      ; PVD through EXTI Line detect
                DCD     RTC_IRQHandler             ; RTC through EXTI Line
                DCD     FLASH_IRQHandler           ; FLASH
                DCD     RCC_CRS_IRQHandler         ; RCC and CRS
                DCD     EXTI0_1_IRQHandler         ; EXTI Line 0 and 1
                DCD     EXTI2_3_IRQHandler         ; EXTI Line 2 and 3
                DCD     EXTI4_15_IRQHandler        ; EXTI Line 4 to 15
                DCD     TSC_IRQHandler             ; TS
                DCD     DMA1_Channel1_IRQHandler   ; DMA1 Channel 1
                DCD     DMA1_Channel2_3_IRQHandler ; DMA1 Channel 2 and Channel 3
                DCD     DMA1_Channel4_5_6_7_IRQHandler ; DMA1 Channel 4, Channel 5, Channel 6 and Channel 7
                DCD     ADC1_COMP_IRQHandler       ; ADC1, COMP1 and COMP2
                DCD     TIM1_BRK_UP_TRG_COM_IRQHandler ; TIM1 Break, Update, Trigger and Commutation
                DCD     TIM1_CC_IRQHandler         ; TIM1 Capture Compare
                DCD     TIM2_IRQHandler            ; TIM2
                DCD     TIM3_IRQHandler            ; TIM3
                DCD     TIM6_DAC_IRQHandler        ; TIM6 and DAC
                DCD     TIM7_IRQHandler            ; TIM7
                DCD     TIM14_IRQHandler           ; TIM14
                DCD     TIM15_IRQHandler           ; TIM15
                DCD     TIM16_IRQHandler           ; TIM16
                DCD     TIM17_IRQHandler           ; TIM17
                DCD     I2C1_IRQHandler            ; I2C1
                DCD     I2C2_IRQHandler            ; I2C2
                DCD     SPI1_IRQHandler            ; SPI1
                DCD     SPI2_IRQHandler            ; SPI2
                DCD     USART1_IRQHandler          ; USART1
                DCD     USART2_IRQHandler          ; USART2
                DCD     USART3_4_IRQHandler        ; USART3 & USART4
                DCD     CEC_CAN_IRQHandler         ; CEC and CAN
                DCD     USB_IRQHandler             ; USB

__Vectors_End
```

Рис. 8. Название в CMSIS функций-обработчиков прерываний для микроконтроллера STM32F072RBT

- *Разрешить генерацию прерывания функциональным модулем* при возникновении определенных событий. Например, разрешить прерывание по событиям завершения передачи и приема USART1 битового пакета, можно следующим образом:

  USART1->CR1 |= USART_CR1_TCIE | USART_CR1_RXNEIE;

- *Разрешить требуемые прерывания в NVIC*, т.е. с помощью регистра **ISER** установить в «1» биты тех прерываний, которые будут обрабатываться в системе. Номер бита определяется с помощью таблицы, показанной на рисунке 2. Например, для разрешения прерываний от модуля USART1 следует установить 27 бит регистра ISER командой:

  NVIC->ISER[0] |= 0x08000000;

- *Разрешить прерывания глобально в микроконтроллере.* Для чего в функции **main()** следует вызвать **__enable_irq()**;

2. Задание

1. Создайте папку «LabHard(<фамилия на латинице>-<номер группы>)-3». Например, студент Иванов из группы 1191б должен создать папку: LabHard(Ivanov-1191)-3.
2. Внутри папки «LabHard…» создайте папки: «Task3_1», «Task3_2» и «Task3_3». В них далее следует размещать проекты среды Keil µVision, соответственно предназначенные для решения: обучающей, самостоятельной и индивидуальных частей задания.

2.1. Обучающая часть

Задача «Interrupt-NVIC»: разработать программные средства для микроконтроллера лабораторного комплекса STM_01, которые реализуют синхронный контроль частоты модуляции светодиода микропереключателями и осуществляют с помощью USART асинхронный прием и преобразование ASCII-кода клавиши, нажатой на удаленном терминале, в последовательность возвращаемых символов.

1. В Keil µVision создайте новый проект и сохраните его в папке Task3_1. Название проекта задайте по следующему шаблону: «<инициалы ФИО на латинице>3_1». Например, проект Сидорова Ивана Петровича должен иметь название: **sip3_1**.
2. Добавьте в проект файл **main.h** и файл программного кода **main.c**.
3. В заголовочный файл добавьте код:

```
#include "stm32f0xx.h"              //Подключение биб-ки с моделью stm32f0xx
void InitUSART1(void);              //Декларация функции инициализации USART1
void USART1_IRQHandler(void);       /*Декларация функции обработки прерывания
                                    от USART1*/
//Функция задержки: count – кол-во периодов задержки длительностью 2.5 мкс
void delay(uint32_t);
```

4. В начале файла **main.c** подключите заголовочный файл **main.h**. Перед описанием функции main() добавьте глобальные переменные, которые будут использоваться в main() и в обработчике прерываний USART – USART_IRQHandler() (рис. 9-а):

   static uint8_t buf[256]; //Буфер данных, передаваемых на ПК посредством USART

   *static uint32_t iReadyTX, iCompleteTX; /*Количество битовых пакетов готовых для передачи и переданных на ПК соответственно*/*

   Модификатор **static** в Keil µVision используется для описания глобальных переменных.

5. Перенесите код функции main(), представленный на рисунке 9-а, в файл **main.c** Вашего проекта. В коде main() можно выделить две части. Сначала осуществляется настройка аппаратных средств микроконтроллера, а затем в цикле while производится синхронный опрос состояния микропереключателей SW3 и SW4 и реализуется программная задержка с помощью функции delay() после включения и выключения светодиода.

   Настройка аппаратных средств микроконтроллера начинается с глобального запрета прерываний в системе с помощью функции __disable_irq(). После вызова этой функции приостановить работу функции main() не способен ни один из обработчиков прерывания и можно считать, что участок кода до вызова __enable_irq() выполняется атомарно. В проекте «Interrupt-NVIC» этот участок кода содержит включение тактирования и настройку режимов работы линий порта B, инициализацию глобальных переменных iReadyTX, iCompleteTX, инициализацию функционального модуля USART1 и разрешение прерывания от USART1 в NVIC.

   Вторая часть main() включает инициализацию переменной с временем базовой задержки и цикл управления работой светодиода. Реализация цикла аналогична той, что представлена в проекте «Task1_1» лабораторной работы №1.

6. Перенесите код функции InitUSART1(), представленный на рисунке 9-б, в файл **main.c** Вашего проекта. Реализация этой функции аналогична той, что представлена в проекте «Task2_1» лабораторной работы №2 за исключением одной строки:

   USART1->CR1 |= USART_CR1_RXNEIE | USART_CR1_TCIE;

   В ней осуществляется конфигурирование USART1 так, чтобы при возникновении событий, приводящих к установке флагов «RXNE» и «TC», модуль микроконтроллера подавал сигнал запроса прерывания в NVIC.

7. Перенесите код функции delay(), представленный на рисунке 9-б, в файл **main.c** Вашего проекта. Реализация этой функции аналогична той, что представлена в проекте «Task1_1» лабораторной работы №1.

8. Перенесите код функции USART1_IRQHandler(), представленный на рисунке 9-в, в файл **main.c** Вашего проекта. Адрес этой функции размещается

библиотекой CMSIS в таблице векторов прерываний NVIC. При поступлении в NVIC сигнала от USART1 он передаст адрес функции USART1_IRQHandler() в счетчик команд микропроцессора и тем самым способствует ее вызову для обработки прерывания.

Тело функции USART1_IRQHandler() содержит два функциональных блока. Первый предназначен для обработки событий, возникающих по завершению приема модулем USART1 битового пакета данных и помещении его в буферный регистр RDR. Второй блок осуществляет обработку событий завершения работы передатчика. Определение типа события, вызвавшего прерывание, производится по флагам «RXNE» и «TC» соответственно.

В первом блоке программного кода сначала выполняется чтение значения битового пакета из буферного регистра приемника USART1 в переменную **pack**:

```
pack=(uint8_t)USART1->RDR;
```

При этом автоматически сбрасывается флаг RXNE в регистре состояния ISR (USART1). Предполагается, что с удаленного терминала приходит ASCII-код нажатой клавиши размером 8 бит. В десятичной системе счисления для представления таких числе достаточно 3 цифр, которые соответствуют сотням, десяткам, единицам. Для определения значений этих цифр и представления каждой цифры ASCII-кодом в USART1_IRQHandler() используется следующий программный код:

```
d100=(uint8_t)pack/100;   //Количество сотен
pack -=d100*100;
//Представление количества сотен ASCII-кодом
buf[(uint8_t)iReadyTX++] = d100+48;
//Количество десятков + преобразование в ASCII-код
buf[(uint8_t)iReadyTX++] = (uint8_t)pack/10+48;
//Количество единиц + преобразование в ASCII-код
buf[(uint8_t)iReadyTX++] = (uint8_t)pack%10+48;
//Символ-разделитель ASCII-кодов клавиш – «пробел»
buf[(uint8_t)iReadyTX++] = 32;
```

В результате его работы на один ASCII-код клавиши, нажатой на удаленном терминале, будет сгенерировано 4 символа (ASCII-кода), из которых три соответствуют десятичным цифрам входной величины, а четвертый является символом-разделителем («пробел»). Эти четыре байта копируются в буферный массив **buf**. Размещение каждого байта сопровождается инкрементом счетчика **iReadyTX**, значение которого показывает полное количество байт, подготовленных для передачи на удаленный терминал.

Нужно отметить, что запись в массив buf является кольцевой. Это позволяет ограничить объем буферного массива 256 байтами. Кольцевая запись реализуется с помощью явного приведения типа uint32_t в uint8_t. Такая операция маскирует (отбрасывает; обращает в ноль) старшие биты переменной iReadyTX (биты с 8 по 31), что ограничивает диапазон изменения

индекса массива buf значениями от 0 до 255. Поэтому байты данных, например, соответствующие iReadyTX==511 или iReadyTX==4095 будет размещаться в ячейке массива buf[255], а соответствующие iReadyTX==512 или iReadyTX==4096 будет размещаться в ячейке массива buf[0].

После записи символов в буферный массив с помощью строк кода:

```
while ((USART1->ISR & USART_ISR_TXE) == 0) {}
USART1->TDR = buf[(uint8_t)iCompleteTX++];
```

инициируется отправка символа в передатчик USART. Она требуется в том случае, когда передатчик USART простаивал, а флаг «TC» был сброшен в обработчике прерывания. Инкремент счетчика iCompleteTX позволяет отслеживать полное количество символов, возвращенных микроконтроллером на удаленный терминал.

Когда передача символа завершается, модуль USART выставляет запрос на прерывание по флагу «TC», который будет обработан во втором блоке функции USART1_IRQHandler(). В этом блоке производится сравнение количества символов, отправленных на удаленный терминал (iCompleteTX), с количеством символов, подготовленных к передаче (iReadyTX). Если iCompleteTX < iReadyTX, то в USART из массива buf копируется следующий символ для отправки на удаленный терминал. В противном случае, осуществляется сброс флага «TC», а передатчик USART переходит в состояние ожидания.

9. Откомпилируйте программу и постройте исполняемый hex-файл.
10. Загрузите hex-файл на микроконтроллер.

Тестирование программы «Interrupt-NVIC» с помощью PuTTY

11. Запустите микроконтроллер в режиме выполнения загруженной программы.
12. Проверьте действие микропереключателей лабораторного комплекса STM_01 на частоту работы светодиода.
13. Запустите на ПК приложение PuTTY, настройте скорость последовательного соединения 115200 бит/с и установите сеанс связи с лабораторным комплексом STM_01.
14. При активном окне терминала PuTTY нажмите различные кнопки на клавиатуре и отследите символьное представление их ASCII-кода. Определите какие последовательности ASCII-кодов соответствуют клавишам со стрелочками, кнопкам «Home», «Page Up», «Esc», «F1 – F12» и др.
15. Определите влияние нажатия клавиш на удаленном терминале на частоту работы светодиода. В том числе, в режиме удержания клавиш и быстрого повторения символов на окне терминала.

```c
/*-------------------------------------------------------------------------------------------------------------------------
**Проект: "Interrupt-NVIC".
**Назначение программы: псевдопараллельное выполнение двух задач:
**                                          * мигание светодиодом с аппаратным контролем частоты переключения;
**                                          * определение ASCII-кодов клавиш, нажимаемых на удаленном терминале;
**Разработчик: Долматов Алексей Викторович - ИЦЭ
**Цель: продемонстрировать структуру программы, использующей внутренние прерывания микроконтроллера STM32F072RBT
**Решаемые задачи:
**        1. Реализация функции-обработчика внутреннего прерывания микроконтроллера STM32F072RBT (USART);
**        2. Конфигурирование NVIC;
**        3. Настройка модуля USART на генерацию сигнала прерывания при возникновении заданных событий.
**-------------------------------------------------------------------------------------------------------------------------*/

#include "main.h"

static uint8_t buf[256];               //Буфер данных, передаваемых на ПК посредством USART
static uint32_t iReadyTX, iCompleteTX;  //Количество битовых пакетов готовых для передачи и переденных на ПК соответственно

int main()
{
  uint32_t half_period, n;      //Половина периода базовой задержки переключения светодиода и степень коэффициента задержки: K=2^n

  __disable_irq();              //Глобальное запрещение прерываний
  //Настройка порта GPIOB для контроля сведодиода
  RCC->AHBENR|=RCC_AHBENR_GPIOBEN;                                  //Включение тактирования порта B: RCC_AHBENR_GPIOBEN=0x00040000
  GPIOB->MODER|=GPIO_MODER_MODER0_0 | GPIO_MODER_MODER8_0;    //Переключение линий 0 и 8 порта B в режим "Output": GPIO_MODER_MODER0_0=0x1; GPIO_MODER_MODER8_0=0x10000
  GPIOB->MODER&=~(GPIO_MODER_MODER12 | GPIO_MODER_MODER13);   //Переключение линий 12(SW4) и 13(SW3) порта B в режим "Input"
  GPIOB->ODR|=0x100;                                                //Разрешение работы светодиодов на стенде STM_01 с помощью установки логической "1" на выводе PB.8
  //---------------------------------------------
  iReadyTX = 0;                 //Сброс количества битовых пакетов, подготовленных для передачи на ПК
  iCompleteTX=0;                //Сброс количества битовых пакетов, переданных на ПК через USART
  InitUSART1();                 //Инициализация модуля USART1
  NVIC->ISER[0] |= 0x08000000;  //Разрешение в NVIC прерывания от модуля USART1
  __enable_irq();               //Глобальное разрешение прерываний

  half_period=50000;            //Инициализация переменной, хранящей половину периода переключения светодиода
  while(1){
    n=((GPIOB->IDR)&0x3000)>>12;                                    //Определение степени коэффициента базовой задержки: K=2^n
    GPIOB->BSRR=0x1;                                                //Зажечь светодиод, подключенный к выводу PB.0
    delay(half_period<<n);                                          //Задержка после  включения светодиода
    GPIOB->BSRR=0x10000;                                            //Погасить светодиод, подключенный к выводу PB.0
    delay(half_period<<n);                                          //Задержка после выключения светодиода
  }
}
```

Рис. 9-а. Проект «Interrupt-NVIC»: основная функция программы – main()

```
//Функция инициализации USART1 микроконтроллера STM32F072RBT
 void InitUSART1(void)
 {
   //Включение тактирования USART1
   RCC->APB2ENR |= RCC_APB2ENR_USART1EN;  //RCC_APB2ENR_USART1EN=0x00004000;
   //Включение тактирования порта A
   RCC->AHBENR |= RCC_AHBENR_GPIOAEN;     //RCC_AHBENR_GPIOAEN=0x00020000;
   //Настройка линий порта A: PA9(TX_1) - выход передатчика; PA10(RX_1) - вход приемника
   GPIOA->MODER |= 0x00280000;            //Перевести линии PA9 и PA10 в режим альтернативной функции
   GPIOA->AFR[1] |= 0x00000110;           //Включить на линиях PA9 и PA10 альтернативную функцию AF1
   //Настройка линии передатчика Tx (PA9)
   GPIOA->OTYPER &= ~GPIO_OTYPER_OT_9;          //Сбросить 9 бит GPIOA->OTYPER - переключение в режим push-pull для линии PA9 (активный выход)
   GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR9;          //Отключение подтягивающих резисторов для линии PA9
   GPIOA->OSPEEDR |= GPIO_OSPEEDR_OSPEEDR9;     //Установка высокой скорости синхронизации линии PA9
   //Настройка линии приемника Rx (PA10)
   GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR10;         //Сброс режима подтягивающих резисторов для линии PA10
   GPIOA->PUPDR |= GPIO_PUPDR_PUPDR10_0;        //Включение подтягивающего резистора pull-up на входной линии PA10 (вход приемника Rx)
   //Конфигурирование USART
   USART1->CR1 &= ~USART_CR1_UE;//Запрещение работы модуля USART1 для изменения параметров его конфигурации
   USART1->BRR = 69;            /*Настройка делителя частоты, тактирующего USART и задающего скорость приема и передачи данных на уровне 115200 бит/с:
                                  Частота тактирующего генератора = 8 МГц; Скорость обмена по USART = 115200 бит/с;
                                  Коэффициент деления = 8000000 / 115200 = 69,4444(4); Округленное значение = 69*/
   USART1->CR1 = USART_CR1_TE | USART_CR1_RE;         /*Разрешить работу приемника и передатчика USART. Остальные биты этого регистра сброшены, что обеспечивает:
                                                        количество бит данных в пакете = 8; контроль четности  - отключен; прерывания по любым флагам USART  - запрещены;
                                                        состояние USART - отключен*/

   USART1->CR1 |= USART_CR1_RXNEIE | USART_CR1_TCIE;  /*Разрешение (в модуле USART1) на выдачу сигнала прерывания при возникновении событий:
                                                        прием кадра в буферный регистр; завершение передачи кадра */

   USART1->CR2 = 0;                                   //Количество стоповых бит - 1
   USART1->CR3 = 0;                                   //DMA1 - отключен
   USART1->CR1 |= USART_CR1_UE;                       //По завершении конфигурирования USART разрешить его работу (биту UE регистра CR1 присвоить 1).
   //Примечание: конфигурирование USART можно выполнять только в не активном состоянии: UE=0.
 }

//Функция задержки: count - количество элементарных периодов задержки с длительностью примерно 2.5 мкс
void delay(uint32_t count)
{
  for (uint32_t i=0;i<count;i++);                    //Выполнение пустых циклов для реализации программной задержки
}
```

Рис. 9-б. Проект «Interrupt-NVIC»: функция инициализации USART1 – InitUSART1() ; программная задержка – delay()

```
89  //Функция-обработчик прерывания от модуля USART1
90  void USART1_IRQHandler(void)
91  {
92    uint8_t pack, d100;
93
94    //Событие готовности принятых данных к чтению
95    if (USART1->ISR & USART_ISR_RXNE) {             //Если в регистре состояний USART1 установлен флаг "RXNE", то
96      pack=(uint8_t)USART1->RDR;                          //Чтение принятого битового пакета из буферного регистра приемника USART1
97      d100=(uint8_t)pack/100;                             //Определение количества сотен в принятом значении
98      pack -=d100*100;                                    //Убрать из принятого значения сотни и оставить в числе 2 значащие цифры, которые соответствуют десяткам и единицам
99      buf[(uint8_t)iReadyTX++] = d100+48;                 //Представить количество сотен в принятом значении в виде ASCII-кода и разместить полученные данные в буфере для передачи на ПК;
100                                                         //увеличить количество данных готовых для передачи на единицу
101     buf[(uint8_t)iReadyTX++] = (uint8_t)pack/10+48;     //Представить количество десятков в принятом значении в виде ASCII-кода и разместить полученные данные в буфере для передачи на ПК;
102                                                         //увеличить количество данных готовых для передачи на единицу
103     buf[(uint8_t)iReadyTX++] = (uint8_t)pack%10+48;     //Представить количество единиц в принятом значении в виде ASCII-кода и разместить полученные данные в буфере для передачи на ПК;
104                                                         //увеличить количество данных готовых для передачи на единицу
105     buf[(uint8_t)iReadyTX++] = 32;                      //Поместить после цифр принятого значения разделитель - "пробел";
106                                                         //увеличить количество данных готовых для передачина единицу
107     while ((USART1->ISR & USART_ISR_TXE) == 0) {}       //Дождаться готовности передатчика USART1 к приему битового пакета для отправки на ПК
108     USART1->TDR = buf[(uint8_t)iCompleteTX++];          //Отправить старшую цифру ASCII-кода в передатчик USART1; увеличить количество переданных на ПК данных на единицу
109   }
110
111   //Событие завершение передачи битового пакета
112   if (USART1->ISR & USART_ISR_TC) {               //Если в регистре состояний USART1 установлен флаг "TC", то
113     // Сброс флага завершения передачи кадра
114     USART1->ICR=USART_ICR_TCCF;                         //Сбросить флаг завершения передачи кадра, чтобы прерывание не сработало повторно
115     //Если количество переданных данных меньше, чем количество подготовленных для передачи, то передать следующий битовой пакет из программного буфера в USART1 для отправки на ПК
116     if (iCompleteTX<iReadyTX){
117       USART1->TDR = buf[(uint8_t)iCompleteTX++];
118     }
119   }
120 }
121
122 /*-------------------------------------------------------------------------------------------------------------------
123   **Руководство пользователя:
124   **       1. Запустите программу на лабораторном комплексе STM_01;
125   **       2. В управлении частотой мигания светодиода используйте микропереключатели SW3(старший разряд) и SW4(младший разряд).
126   **          Примерная частота мигания: 00 - 4 Гц; 01 - 2 Гц; 10 - 1 Гц; 11 - 0.5 Гц;
127   **       3. На компьютере запустите приложение PuTTY и подключитесь к соответствующему COM-порту на скорости 115200 бит/с;
128   **       4. При активном окне терминала нажмите различные кнопки клавиатуры и отследите ASCII-код каждой клавиши
129   **          (управляющим клавишам должны соответствовать последовательности ASCII-кодов);
130   **-------------------------------------------------------------------------------------------------------------------*/
```

Рис. 9-в. Проект «Interrupt-NVIC»: обработчик прерывания модуля USART1 – USART1_IRQHandler()

2.2. Самостоятельная работа

Цель Task3_2: асинхронное управление частотой мигания светодиода из приложения PuTTY.

В процессе разработки программных средств для взаимодействия лабораторного комплекса STM_01 с удаленным терминалом необходимо решить следующие задачи:

1. Частота мигания светодиода должны быть переменным параметром, задаваемым с помощью приложения PuTTY.
2. При включении микроконтроллера светодиод должен мигать с частотой, заданной по умолчанию.
3. При подключения приложения PuTTY к лабораторному комплексу STM_01 и нажатию клавиши «Enter» в окне терминала должно отобразиться текущее значение периода мигания светодиода в микросекундах и приглашение для ввода нового значения (рис. 10).

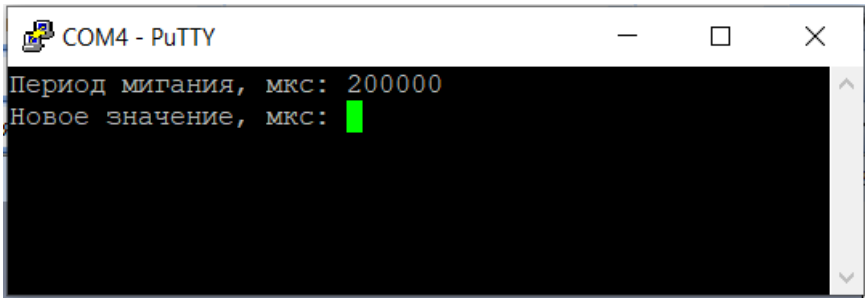

Рис. 10. Примерный вид окна PuTTY

4. Обработчик прерывания USART1 должен обеспечивать прием от удаленного терминала только цифр, символов «enter» и «backspace». ASCII-коды других клавиш должны игнорироваться обработчиком прерывания.

   Примечание. Для перемещения между строками в окне терминала PuTTY используйте ASCII-последовательности клавиш «стрелка вверх» и «стрелка вниз», отправляемые микроконтроллером. Чтобы определить ASCII-последовательности этих клавиш воспользуйтесь проектом «Interrupt-NVIC».

5. Программные средства микроконтроллера должны позволять вводить не более 6 цифр периода мигания светодиода. Для корректировки ранее введенных цифр в окне терминала PuTTY следует использовать клавишу «backspace».
6. Новое значение периода мигания должно применяться только после нажатия клавиши «enter» в окне PuTTY, при условии, что введена хотя бы одна цифра этой величины.

2.3. Индивидуальное задание

Цель Task3_3: «программный отладчик» – функция, которая добавляется в код тестируемой программы для приостановки ее выполнения, реагирует на команды удаленного терминала и позволяет отображать в нем текущие значения переменных тестируемой программы.

В проекте должны быть решены следующие задачи:

1. В соответствие с вариантом задания (**Приложение 1**) реализуйте в main() алгоритм итерационных вычислений на основе арифметической или геометрической прогрессии.
2. Определите глобальную статическую переменную для хранения флагов, показывающих прием команд от удаленного терминала и их выполнение. Статусная переменная должна содержать как минимум флаги соответствующие следующим событиям, произошедшим на удаленном терминале PuTTY:
   - нажата клавиша «F5»;
   - нажата клавиша «i»;
   - нажата клавиша «e»;
   - нажата клавиша «s».
3. В обработчике прерывания USART1 реализуйте **установку** флагов в статусной переменной при приеме от терминала PuTTY кодов следующих клавиш (команд): «F5», «i», «e», «s».
4. Разработайте в проекте функцию **debug()**, которая будет захватывать процессор микроконтроллера до момента установки флага, соответствующего нажатию на терминале PuTTY клавиши «F5». Кроме того, функция debug() должна отслеживать появление всех остальных указанных выше флагов и производить соответствующую каждому флагу обработку:
   - «i» - вывести в окне терминала PuTTY значение переменной с номером итерации вычислений;
   - «e» - вывести в окне терминала PuTTY значение переменной с элементом вычислений, соответствующим текущей итерации;
   - «s» - вывести в окне терминала PuTTY текущее значение переменной с накапливаемым результатом вычислений.

   По завершении обработки соответствующий флаг должен быть **сброшен** в статусной переменной функцией debug().
5. Поместите функцию debug() в точку останова основной программы и протестируйте работу программного отладчика.

3. Подготовка отчета, представление и оценка работы

Комментарии в программном коде в качестве «Отчета о выполнении задания»

1. Комментарии в программном коде каждого проекта выступают в роли отчета и оцениваются в процессе защиты работы.
2. Структура комментариев, описывающих программный код должна быть следующей:
   - В начале файла «main.c» должны присутствовать сведения о разработчике (ФИО - группа), цели выполнения работы и задачах, которые решались в ходе создания программы;
   - Перед реализацией каждой функции следует указать ее назначение и привести список входных/выходных параметров с описанием;
   - Внутри тела функции следует выделить отдельные блоки, реализующие единую комплексную операцию обработки данных, и вынести в их заголовок сведения о назначении блока;
   - Каждая элементарная операция в строке кода должна быть снабжена комментарием;
   - В конце файла «main.c» нужно привести комментарии, содержащие краткое руководство использования программы на стенде STM_01.
3. В заголовочных файлах (с расширением «h»), созданных разработчиком проекта, следует прокомментировать все строки кода. Однако можно опустить заголовочный и финальный комментарии со сведениями о проекте и руководстве пользователя.

Представление и защита работы

4. Для каждой части **Задания** нужно построить отдельный проект Keil µVision, протестировать и отладить его работу на стенде STM_01, снабдить программный код комментариями в соответствие с п. 3.1- 3.3.
5. Представление работы начинается с демонстрации преподавателю работоспособности программного кода проекта на стенде STM_01. Каждый проект (часть задания лабораторной работы) может быть представлена отдельно. В ходе представления преподаватель оценивает функциональность программы и качество написания программного кода. Студент должен быть готов ответить на вопросы преподавателя по использованным в проекте алгоритмам и их программной реализации.
6. После получения оценки за все проекты лабораторной работы следует создать единый архив (со всеми проектами), и загрузить его на сайт Eluniver.
7. По комментариям к проекту преподаватель оценивает качество отчета по каждому из выполненных заданий.
8. После представления всех заданий лабораторной работы студент получает право ее защиты, которая заключается в ответе на 2 контрольных вопроса из

обозначенного в **разделе 4** списка. Каждый контрольный вопрос оценивается отдельно.

Структура оценки лабораторной работы

№	Вид оценки	Максимальный балл
1.	Проект «Обучающая часть»	10
2.	Проект «Самостоятельная работа»	15
3.	Проект «Индивидуальное задание»	25
4.	Отчет «Обучающая часть»	5
5.	Отчет «Самостоятельная работа»	5
6.	Отчет «Индивидуальное задание»	10
7.	Контрольный вопрос 1	15
8.	Контрольный вопрос 2	15
	Итого:	**100**

4. Контрольные вопросы

1. Что понимается под функциональным блоком микроконтроллера? Назовите не менее 5 функциональных блоков и поясните их назначение.
2. Какие аппаратные события Вам известны?
3. Каково назначение флагов?
4. Каким образом можно узнать о состоянии того или иного флага?
5. Что понимается под аппаратным прерыванием?
6. В чем различие между аппаратным событием и аппаратным прерыванием?
7. Какие аппаратные события характерны для модуля USART?
8. При возникновении какого события устанавливается флаг «RXNE»?
9. Какое событие маркируется в модуле USART маркируется флагом «TXE»?
10. Как осуществляется сброс флага «TC»?
11. Какой флаг в модуле USART устанавливается по завершению передачи битовых данных в линию связи?
12. Когда аппаратное событие порождает аппаратное прерывание?
13. Сколько различных сигналов прерываний может генерировать один периферийный модуль микроконтроллера?
14. Какой регистр USART используется для разрешение прерываний по тем или иным аппаратным событиям?
15. В чем различие между внутренними и внешними прерываниями?
16. Как называется контроллер, отвечающий за обработку внешних прерываний?
17. В чем различие между прерываниями и исключениями?
18. Сколько линий прерываний обслуживает NVIC?
19. Какой регистр NVIC позволяет разрешать прерывания от того или иного периферийного модуля микроконтроллера?
20. К какому функциональному блоку относится регистр ISPR?
21. Каково назначение регистра ICER?
22. К какой линии NVIC подключен модуль USART1?
23. Что понимается под таблицей векторов прерываний?
24. Как приоритет прерываний влияет на их обработку?
25. Что представляет собой обработчик прерываний? Поясните на примере.
26. Как узнать наименование обработчиков прерываний в библиотеке CMSIS для микроконтроллера STM32F072RBT?
27. Каким образом осуществляется настройка приоритета прерываний?
28. Как определяется номер регистра IPR и расположение в нем управляющих бит для функционального модуля микроконтроллера?
29. Можно ли запретить любые прерывания при работе микроконтроллера?
30. Какие шаги являются обязательными при разработке программных средств микроконтроллера, использующих прерывания?
31. Какой ASCII-код соответствует кнопке «end»?
32. Каким образом в PuTTY можно организовать многострочный вывод символом (текста) в окне терминала?

Приложение 1. Варианты индивидуальных заданий

Вариант	Описание задания
1.	Найдите сумму 15 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 4$, $d = 2$
2.	Найдите сумму 11 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 2$, $q = 3$
3.	Найдите произведение 17 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 26$, $d = -1$
4.	Найдите произведение 9 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 7$, $q = -5$
5.	Найдите сумму 16 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = -5$, $d = 6$
6.	Найдите сумму 20 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 7$, $q = -5$
7.	Найдите произведение 12 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = -50$, $d = 3$
8.	Найдите произведение 19 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = -3$, $q = -7$
9.	Найдите сумму 13 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 11$, $d = 16$
10.	Найдите сумму 14 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 7$, $q = -5$
11.	Найдите произведение 7 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 30$, $d = 5$
12.	Найдите произведение 8 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 23$, $q = -15$
13.	Найдите сумму 25 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 0$, $d = -4$
14.	Найдите сумму 14 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = -7$, $q = 12$
15.	Найдите произведение 10 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = -79$, $d = 15$
16.	Найдите произведение 8 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = -9$, $q = 4$
17.	Найдите сумму 23 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 4$, $d = 18$
18.	Найдите сумму 34 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 9$, $q = -25$
19.	Найдите произведение 17 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 356$, $d = -29$
20.	Найдите произведение 35 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 2$, $q = -7$
21.	Найдите сумму 6 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 402$, $d = -117$
22.	Найдите сумму 24 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = -19$, $q = 5$
23.	Найдите произведение 32 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = -45$, $d = 16$
24.	Найдите произведение 8 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 68$, $q = -13$
25.	Найдите сумму 16 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 85$, $d = 114$
26.	Найдите сумму 10 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = -153$, $q = 69$
27.	Найдите произведение 17 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = -511$, $d = 46$
28.	Найдите произведение 36 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 8$, $q = -5$
29.	Найдите сумму 15 членов арифметической прогрессии $a_{n+1} = a_n + d$, где $a_1 = 31$, $d = -9$
30.	Найдите сумму 29 членов геометрической прогрессии $a_{n+1} = a_n \cdot q$, где $a_1 = 39$, $q = -3$