|  |
| --- |
| **Отчет по лабораторной работе №5****«Работа с АЦП»** |
| Дата | Оценка(max 5) | Бонус засложность | Подпись |

**Цели работы:**

* + ознакомиться с модулем встроенного АЦП.

**Задачи работы:**

 - разработать программу, передающую результаты измерения в вольтах на ПК по интерфейсу UART.

**Краткий конспект теоретической части**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  **АЦП. Виды. Характеристики.**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| **Основные регистры встроенного АЦП** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

 |

**Структурная схема АЦП (ADC1)**

****

**Временная диаграмма работы АЦП (ADC1)**

****

**Задание 1.** Разработать программу, передающую результаты измерения с трех каналов АЦП в вольтах на ПК по интерфейсу UART.

Листинг кода:

|  |
| --- |
| **#include** "stm32f10x.h"**void** **USART1\_IRQHandler** (**void**); //Объявляем обработчик прерывания по USART1**int** **main**(**void**){ **int** i; RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN; //тактируем PORTA RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN; //тактируем АЦП GPIOA->CRL&=~GPIO\_CRL\_CNF0\_1; //Настраиваем пин PA0 на GPIOA->CRL&=~GPIO\_CRL\_CNF0\_0; //альтернативную функцию, как вход GPIOA->CRL&=~GPIO\_CRL\_MODE0\_1; GPIOA->CRL&=~GPIO\_CRL\_MODE0\_0; GPIOA->CRL&=~GPIO\_CRL\_CNF1\_1; //Настраиваем пин PA1 на GPIOA->CRL&=~GPIO\_CRL\_CNF1\_0; //альтернативную функцию, как вход GPIOA->CRL&=~GPIO\_CRL\_MODE1\_1; GPIOA->CRL&=~GPIO\_CRL\_MODE1\_0; GPIOA->CRL&=~GPIO\_CRL\_CNF2\_1; //Настраиваем пин PA2 на GPIOA->CRL&=~GPIO\_CRL\_CNF2\_0; //альтернативную функцию, как вход GPIOA->CRL&=~GPIO\_CRL\_MODE2\_1; GPIOA->CRL&=~GPIO\_CRL\_MODE2\_0; ADC1->SMPR2|= ADC\_SMPR2\_SMP0;//Задаем макс. время выборки для 0 канала ADC1->SQR1 &= ~ADC\_SQR1\_L; //Задаем одну конверсию ADC1->CR2 |= ADC\_CR2\_ADON; /Включяаем АЦП **for**(i = 0; i < 10000; i++ ); //Задержка, время на стабилизацию  RCC->APB2ENR|=RCC\_APB2ENR\_USART1EN; // Тактирование USART GPIOA->CRH |= GPIO\_CRH\_CNF9\_1; //Настраиваем на выход PA9 Tx GPIOA->CRH &= ~GPIO\_CRH\_CNF9\_0; GPIOA->CRH &= ~GPIO\_CRH\_MODE9\_1; GPIOA->CRH |= GPIO\_CRH\_MODE9\_0; GPIOA->CRH &= ~GPIO\_CRH\_CNF10\_1; //Настраиваем на вход PA10 Rx GPIOA->CRH |= GPIO\_CRH\_CNF10\_0; GPIOA->CRH &= ~GPIO\_CRH\_MODE10\_1; GPIOA->CRH &= ~GPIO\_CRH\_MODE10\_0; USART1->BRR = ((8000000L/(16L\*9600L))<<4);//Задание предделителя USART1->CR1 |= USART\_CR1\_UE; //Включение USART USART1->CR1 &= ~USART\_CR1\_M; //Длина слова 8 бит USART1->CR1 |= USART\_CR1\_TE; //Включаем передатчик USART1->CR2 &= ~USART\_CR2\_STOP; /1 стоп-бит USART1->CR1 |= USART\_CR1\_RE; /Включаем приемник USART1->CR1 |= USART\_CR1\_RXNEIE; //Включаем прерывание по приему \_\_enable\_irq(); //Делаем доступными прерывания NVIC\_SetPriority(*USART1\_IRQn*, 1); //Устанавливаем приоритет для прерывания NVIC\_EnableIRQ(*USART1\_IRQn*); //Включаем прерывание от USART1 **while**(1) { }}**void** **USART1\_IRQHandler** (**void**) //Обработчик прерывания по USART1{ uint8\_t i = 0; uint8\_t buffer[2] = {0,0}; uint16\_t \*buf = &buffer; **char** vars[6] = "M0M1M2"; uint16\_t \*M0 = &vars[0]; uint16\_t \*M1 = &vars[2]; uint16\_t \*M2 = &vars[4]; **int** mV; **for**(i=0;i<2;i++){ **while** (!(USART1->SR&USART\_SR\_RXNE)); buffer[i] = USART1->DR; //Копируем их в буфер } **if** (\*buf == \*M0){ ADC1->SQR3 &= ~ADC\_SQR3\_SQ1;//Задаем 0 канал для конверсии  ADC1->CR2 |= ADC\_CR2\_ADON; //Запускаем конверсию **while** (!(ADC1->SR&ADC\_SR\_EOC)); //Ждем 1 в бите завершения конверсии mV = (3300 \* (ADC1->DR)) / 4096; printf("%u mV\n",(**int**)(mV)); } **if** (\*buf == \*M1){ ADC1->SQR3 = 0; ADC1->SQR3 |= ADC\_SQR3\_SQ1\_0;//Задаем 1 канал для конверсии  ADC1->CR2 |= ADC\_CR2\_ADON; //Запускаем конверсию **while** (!(ADC1->SR&ADC\_SR\_EOC)); //Ждем 1 в бите завершения конверсии mV = (3300 \* (ADC1->DR)) / 4096; printf("%u mV\n",(**int**)(mV)); } **if** (\*buf == \*M2){ ADC1->SQR3 = 0; ADC1->SQR3 |= ADC\_SQR3\_SQ1\_1;//Задаем 2 канал для конверсии ADC1->CR2 |= ADC\_CR2\_ADON; //Запускаем конверсию **while** (!(ADC1->SR&ADC\_SR\_EOC)); //Ждем 1 в бите завершения конверсии mV = (3300 \* (ADC1->DR)) / 4096; printf("%u mV\n",(**int**)(mV)); } USART1->SR &= ~USART\_SR\_RXNE;} |

|  |
| --- |
| **Запишите ваши конкретные выводы:** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |