ИСПОЛЬЗОВАНИЕ ВСТРОЕННЫХ ЦИФРО-АНАЛОГОВОГО И АНАЛОГО-ЦИФРОВОГО ПРЕОБРАЗОВАТЕЛЕЙ МИКРОКОНТРОЛЛЕРА STM32F4DISCOVERY

Квашнин В. О., Бабаш А. В.

Рассмотрены основные возможности цифро-аналоговых и аналого-цифровых встроенных преобразователей микроконтроллера STM32F4Discovery. Приведены области применения в технике. Приведены особенности настройки и программирования встроенных цифро-аналоговых и аналого-цифровых преобразователей при помощи языка программирования высокого уровня С. Представлены конкретные примеры использования цифро-аналоговых преобразователей для формирования выходного сигнала любой формы. Приведен конкретный пример использования аналого-цифрового преобразователя для обработки и визуализации преобразованного в цифровой вид сигнала на входе аналого-цифрового преобразователя.

Розглянуто основні можливості цифро-аналогових та аналого-цифрових вбудованих перетворювачів мікроконтролера STM32F4Discovery. Наведені області використання у техніці. Наведено особливості налаштування та програмування вбудованих цифро-аналогових та аналогово-цифрових перетворювачів за допомогою мови програмування високого рівня С. Представлені конкретні приклади використання цифро-аналогових перетворювачів для формування вихідного сигналу будь-якої форми. Наведено конкретний приклад використання аналого-цифрового перетворювача для обробки та візуалізації перетвореного у цифровий вигляд сигналу на вході аналого-цифрового перетворювача.

The main capabilities of digital-to-analog and analog-to-digital built-in converters of STM32F4discovery microcontroller are considered here. Areas of their using in technics are given. Adjustment and programming features of digital-to-analog and analog-to-digital converters by means of high-level programming language C are given. Concrete examples of using digital to analog converters to form output signal of any form are considered. Concrete example of using analog to digital converter for processing and visualizing transformed into digital form signal from analog to digital converter input is given.

Квашнин В. О. канд. техн. наук, доц. каф. ЭСА ДГМА
tm@dgma.donetsk.ua
Бабаш А. В. аспирант каф. ЭСА ДГМА

ДГМА – Донбасская государственная машиностроительная академия, г. Краматорск.

УДК 681.3

Квашнин В. О., Бабаш А. В.

ИСПОЛЬЗОВАНИЕ ВСТРОЕННЫХ ЦИФРО-АНАЛОГОВОГО И АНАЛОГО-ЦИФРОВОГО ПРЕОБРАЗОВАТЕЛЕЙ МИКРОКОНТРОЛЛЕРА STM32F4DISCOVERY

Быстрое развитие цифровой техники и цифровых методов обработки сигналов определяет современные тенденции в разработке самых разнообразных устройств, приборов и систем. При этом значительная роль принадлежит аналого-цифровым и цифро-аналоговым преобразователям (АЦП и ЦАП). Они широко используются во всех областях радиоэлектроники, различной измерительной и контрольной аппаратуре, системах связи и т.д. [1, 2].

Цели и задачи. Определение возможностей ЦАП и АЦП микроконтроллера STM32F4Discovery, особенностей его программирования на языке высокого уровня – С, а также разработка методики и алгоритма настройки и работы ЦАП и АЦП.

Современные микроконтроллеры, такие как STM32, Simatic S7-200, S7-300, PIC контроллеры и т.д. содержат в себе встроенные цифро-аналоговые и аналогово-цифровые преобразователи. Аналоговые входы микроконтроллеров широко используются в промышленности для захвата аналогового сигнала, например, с датчика температуры с его последующей обработкой (преобразованием в цифровую форму) и выводом на экран монитора для отображения его текущего значения и соответственно контроля этого параметра [3–7].

Цифро-аналоговые преобразователи активно используются для формирования выходного сигнала нужной формы на основе цифрового массива данных.

Для исследования возможностей встроенных ЦАП и АЦП современных микроконтроллеров выбран микроконтроллер STM32F4Discovery (рис.1).

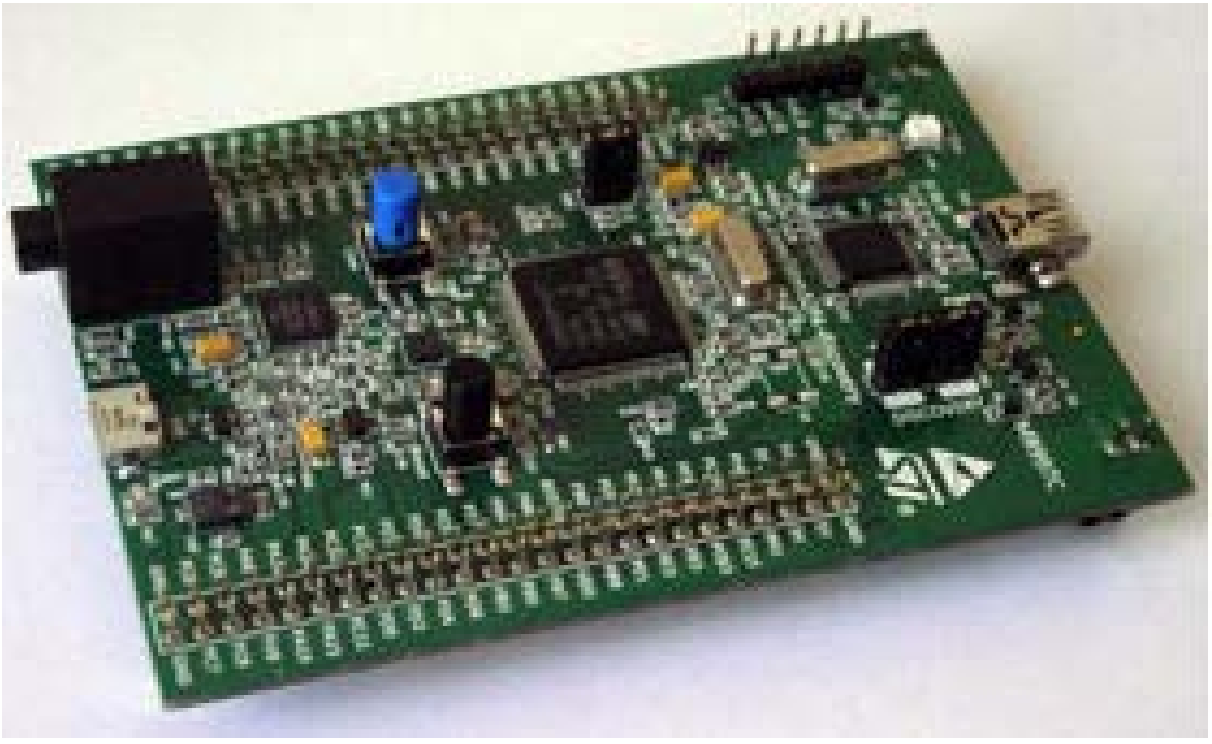

Рис. 1. Микроконтроллер STM32F4discovery

Характеристики микроконтроллера STM32F4Discovery 168 МГц, 1 Мб Flash, 192 Кб ОЗУ, ST-LINK / V2, 3-х осевой акселерометр, цифровой микрофон, USB-OTG, 24-разрядный аудио ЦАП с усилителем класса D, 8 светодиодов, 2 кнопки .

Микроконтроллер STM32F407VG включает в себя три АЦП. Разрядность всех АЦП составляет 12 бит. Каждый преобразователь способен принимать сигнал с шестнадцати внешних каналов. Кроме того, в состав контроллера входит датчик температуры.

Диапазон входных напряжений составляет 1,8...3,6 В. Датчик температуры подключен к входному каналу ADC_IN16, который используется для того, чтобы преобразовать выходное напряжение датчика в цифровое значение [8].

Внутренний датчик температуры предназначен для отслеживания изменения температуры, а не для ее измерения, поскольку смещение показателей датчика может изменяться в ходе изменений параметров процесса. Поэтому, если необходимо точное измерение абсолютных значений температуры, то для этого лучше использовать внешний датчик.

Оценка возможностей микроконтроллера может быть продемонстрирована посредством захвата входного аналогового сигнала с выхода ЦАП (PA5), с последующим его превращением в цифровую форму и его передачей на персональный компьютер (ПК) с помощью преобразователя USB-UART с отображением графика изменения входного сигнала на экране монитора ПК.

Схема подключения выводов ЦАП (PA5) ко входу АЦП (PA1) микроконтроллера приведена на рис. 2. При этом необходимо замкнуть выводы PA5 и PA1.

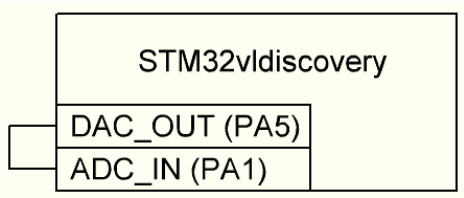

Рис. 2. Схема подключений для демонстрации возможностей встроенного АЦП

Для передачи кодов АЦП и микроконтроллера на ПК используется преобразователь USB-UART (Universal asynchronous receiver transmitter). При подключении к ПК и установке специальной программы (драйвера), преобразователь определяется операционной системой как последовательный коммуникационный порт COM. Теперь любая терминальная программа на ПК может принимать и передавать данные через USB-UART.

Схема подключения преобразователя USB-UART к микроконтроллеру выводам UART3 микроконтроллера STM32F4Discovery приведена на рис. 3.

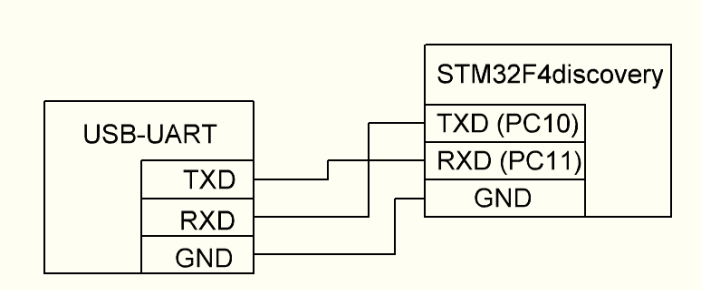

Рис. 3. Схема подключений USB-UART к микроконтроллеру

В качестве входного аналогового сигнала на входе АЦП (PA1) используется выход ЦАП (PA5). Порт PA1 микроконтроллера STM32F4Discovery можно использовать как вход АЦП (ADC1 канал 1). Нужную информацию можно найти в документации к STM32F4Discovery (рис. 4) [9, 10].

PA1	USART2_RTS/ USART4_RX/ ETH_RMII_REF_CLK/ ETH_MII_RX_CLK/ TIM5_CH2/ [illegible] ADC123_IN1	24	-	-	-	-
	[illegible] TIM5_CH3/					

Рис. 4. Порт АЦП

На вход АЦП поступает аналоговый сигнал пилообразной формы, сформированный в микроконтроллере на выходе ЦАП. График изменения кодов АЦП входного сигнала представлен на рис. 5 в виде разработанного приложения для визуализации сигналов в среде Delphi с использованием свободно-распространяемого компонента (набора библиотек) BComPort, который позволяет работать с последовательным портом COM.

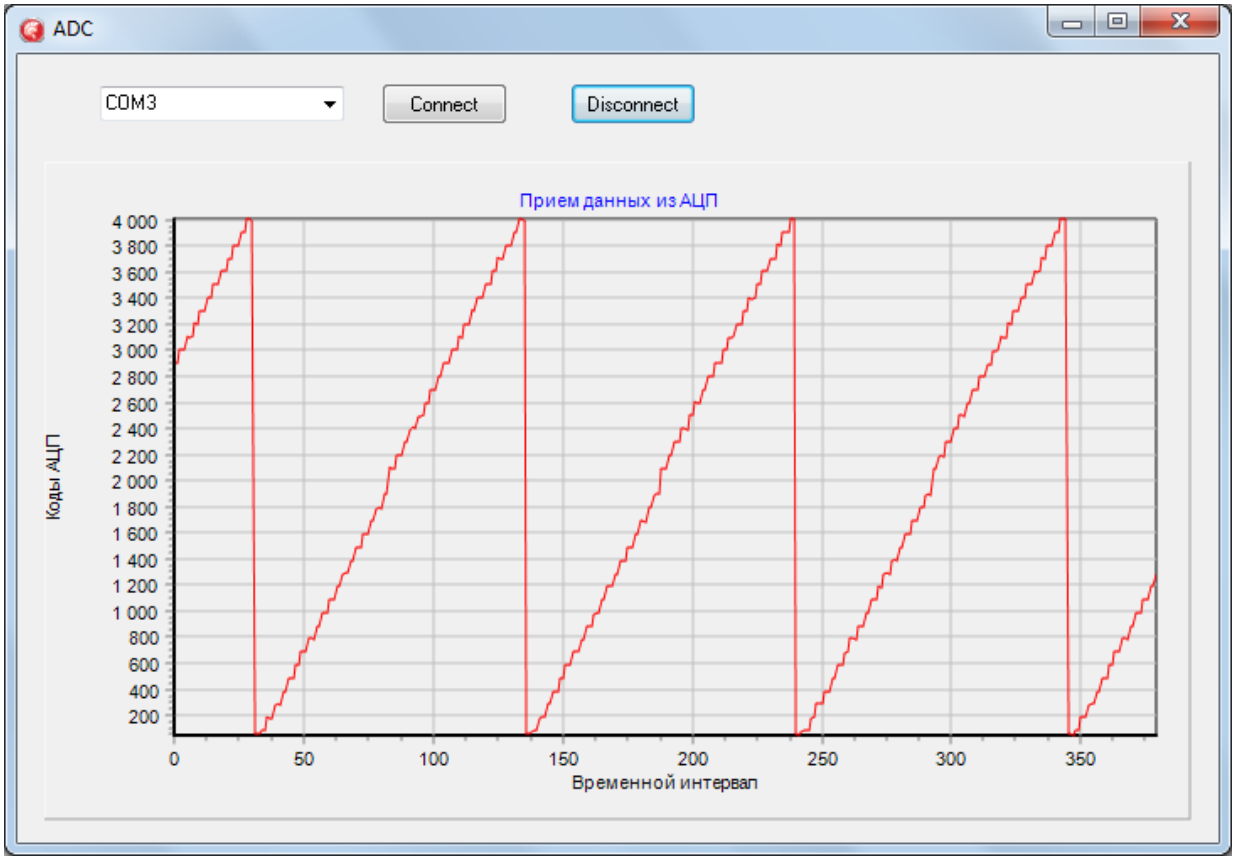

Рис. 5. Разработанное приложение для визуализации информации в среде Delphi

Разработанная блок-схема алгоритма настройки имеющегося в микроконтроллере АЦП и его порта представлена на рис. 6.

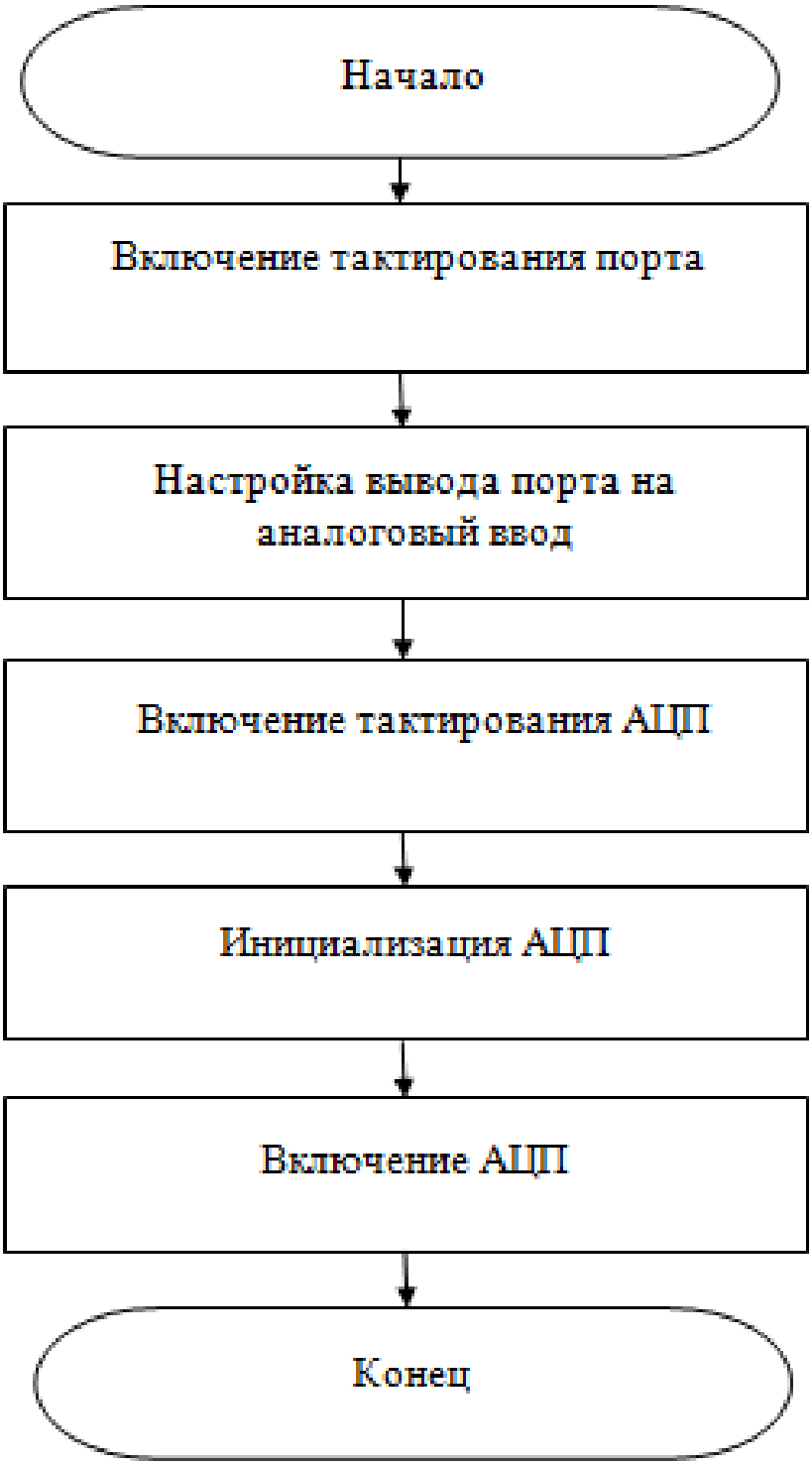

Рис. 6. Блок-схема алгоритма настройки АЦП

Для того чтобы настроить встроенный аналого-цифровой преобразователь, необходимо настроить порт (в данном случае используется PA1) так, чтобы он работал как аналоговый вход АЦП. При настройке выхода порта PA1 на вход необходимо программно выбрать режим аналогового входа. На рис. 7 приведен программный код процедуры настройки и инициализации входа-выхода порта для работы в режиме аналогового входа АЦП ADC1.

```
void adc_pin_init() {

      GPIO_InitTypeDef gpio;

      RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
      GPIO_StructInit(&gpio);
      gpio.GPIO_Mode = GPIO_Mode_AN;
      gpio.GPIO_Pin = GPIO_Pin_1;
      GPIO_Init(GPIOA, &gpio);
      }
```

Рис. 7. Настройка режима работы вывода порта PA1

В данной процедуре сначала происходит включение порта при помощи команды RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE). Далее выбирается режим аналогового входа (gpio.GPIO_Mode = GPIO_Mode_AN), после чего реализуется применение структуры gpio для выполнения настроек порта.

После выполнения процедуры настройки порта выполняется настройка режима АЦП ADC1 для того, чтобы он мог производить измерения и конвертацию сигнала в цифровую форму.

На рис. 8 приведена процедура инициализации ADC1, которая осуществляет его настройку.

```
// Настраиваем АЦП

     void adc_init() {

           ADC_InitTypeDef ADC_InitStructure;
           ADC_CommonInitTypeDef adc_init;
           RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
           ADC_DeInit();
           ADC_StructInit(&ADC_InitStructure);
           adc_init.ADC_Mode = ADC_Mode_Independent;
           adc_init.ADC_Prescaler = ADC_Prescaler_Div2;
           ADC_InitStructure.ADC_ScanConvMode = DISABLE;
           ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
           ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigConvEdge_None;
           ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
           ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
           ADC_CommonInit(&adc_init);
           ADC_Init(ADC1, &ADC_InitStructure);
           ADC_Cmd(ADC1, ENABLE);
           }
```

Рис. 8. Процедура инициализации АЦП

Сначала на ADC1 подаются тактовые импульсы от шины APB2 (команда RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE)). То есть на АЦП подается питание (по умолчанию в целях уменьшения энергопотребления все периферийные устройства микроконтроллера STM32F4Discovery отключены). Команда ADC_DeInit() производит сброс всех настроек. Это делается для перестраховки при инициализации. Далее выбирается независимый режим работы ADC1 (ADC_Mode_Independent). То есть работа ADC1 не зависит от других АЦП микроконтроллера.

С помощью команд ADC_InitStructure.ADC_ScanConvMode = DISABLE и ADC_InitStructure.ADC_ContinuousConvMode = DISABLE выбирается режим работы недлительного и одиночного преобразования ADC1. Команда ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConvEdge_None позволяет выбрать источник запуска преобразований регулярной группы каналов АЦП (в данном случае

не используется). Им может быть какое-либо прерывание или внешнее событие. ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right выравнивает данные по правому краю. ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b выбирает разрешение АЦП 12 бит.

Для того чтобы АЦП ADC1 мог осуществлять какие-либо измерения и преобразования, необходимо его включить (ADC_Cmd(ADC1,ENABLE)). Иначе он просто не будет функционировать.

Существуют инжектированные и регулярные каналы АЦП. Результат преобразований АЦП из регулярных каналов сохраняется в одном регистре.

Инжектированные каналы сохраняют результаты преобразований в отдельные регистры. В данном случае используются регулярные каналы.

Для преобразования аналогового сигнала на входе АЦП разработана специальная функция для считывания и конвертации аналогового сигнала в цифровую форму (рис. 9).

```
// Функция чтения данных из АЦП

      u16 readADC1(u8 channel) {
            ADC_RegularChannelConfig(ADC1, channel, 1, ADC_SampleTime_3Cycles);
            ADC_SoftwareStartConv(ADC1);
            while (ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) == RESET);
            return ADC_GetConversionValue(ADC1);
            }
```

Рис. 9. Функция считывания и конвертации аналогового сигнала в цифровую форму

Команда ADC_RegularChannelConfig(ADC1, channel, 1, ADC_SampleTime_3Cycles) выполняет конфигурацию регулярных каналов АЦП. При конфигурации ADC1, выбирается его канал АЦП, который определяется переменной channel и настраивается время выборки. От параметра ADC_SampleTime зависит точность выполняемых преобразований.

Функция ADC_SoftwareStartConv(ADC1) разрешает преобразования для ADC1. В цикле while (ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC)==RESET) ожидается окончание запущенного процесса конвертации величины аналогового сигнала в цифровую форму.

Строка return ADC_GetConversionValue(ADC1) возвращает текущее значение величины оцифрованного сигнала на входе.

Вызываемая из основной программы функция readADC1() осуществляет процесс преобразования аналогового сигнала на входе АЦП в цифровую форму. Фрагмент программного кода, который выполняет оцифровку аналогового сигнала и его передачу при помощи интерфейса USB-UART представлен на рис. 10. Изменение величины аналогового сигнала сигнализируется четырьмя пользовательскими светодиодами на отладочной плате STM32F4Discovery.

Микроконтроллер STM32F407VG включает в себя два встроенных ЦАП. Разрядность ЦАП составляет 12 бит. ЦАП (DAC), использует в качестве выходов порт А: PA4 - DAC1, PA5 - DAC2 (рис. 11).

Цифро-аналоговые преобразователи используются для конвертации цифровой информации в сигнал аналоговой формы и находят применение для воспроизведения звука, формирования сигнала нужной частоты и величины напряжения.

Встроенный ЦАП может работать в 3-х режимах:

- генерация напряжения по значению, записанному в порт данных;
- генерация пилообразного сигнала;
- генерация белого шума.

```
while(1) //Бесконечный цикл в нем мы проверяем ...

{


      // Читаем данные из АЦП

         bin_code = readADC1(ADC_Channel_1);

      // Зажигаем светодиоды в зависимости от значения bin_code

         if(bin_code>=1000)
         {
               GPIO_SetBits(GPIOD, GPIO_Pin_12);
         }

         else
            GPIO_ResetBits(GPIOD, GPIO_Pin_12);


         if(bin_code>=2000)
         {
               GPIO_SetBits(GPIOD, GPIO_Pin_13);
         }

         else
            GPIO_ResetBits(GPIOD, GPIO_Pin_13);


         if(bin_code>=3000)
         {
               GPIO_SetBits(GPIOD, GPIO_Pin_14);
         }

         else
            GPIO_ResetBits(GPIOD, GPIO_Pin_14);


         if(bin_code>=4000)
         {
               GPIO_SetBits(GPIOD, GPIO_Pin_15);
         }

         else
            GPIO_ResetBits(GPIOD, GPIO_Pin_15);

         Delay(1000000);
// Передаем данные на ПК
         send_to_uart('\r');
         send_to_uart('\n');
         send_int_Uart(USART3,bin_code);
         send_to_uart('\r');
         send_to_uart('\n');

         Delay(1000000);


   }

}
```

Рис. 10. Фрагмент программного кода для оцифровки сигнала и передачи данных через USB-UART

PA4	SPI1_NSS/ SPI3_NSS/ USART2_CK/ DCMI_HSYNC/ OTG_HS_SOF/ I2S3_WS/ ADC12_IN4/ DAC1_OUT	29	LRCK/AIN1x	-	-
PA5	SPI1_SCK/ OTG_HS_ULPI_CK/ TIM2_CH1_ETR/ TIM8_CHIN/ ADC12_IN5/ DAC2_OUT	30	-	-	SCL/SPC

Рис. 11. Каналы ЦАП STM32F4DISCOVERY

Для оценки возможностей встроенного ЦАП использовался сформированный пилообразный сигнал напряжения на его выходе. Формирование сигнала было выполнено на основе массива целых чисел. Так как разрядность ЦАП 12 бит, максимальное число, которое можно записать в регистр ЦАП будет 4095. Это число соответствует максимальному выходному напряжению 3 В. Посредством несложных арифметических операций можно устанавливать уровни напряжений в диапазоне 0..3 В. Значение регистра ЦАП определяется следующим образом

$$DAC_{вых} = \frac{U_{вых} \cdot 4095}{U_{\max}}. \quad (1)$$

где $U_{вых}$ – требуемое значение выходного напряжения; U_{max} – максимальное значение напряжения; $DAC_{вых}$ – значение регистра ЦАП.

При тестировании ЦАП для формирования выходного сигнала использовалось 40 точек. Заполнение регистра ЦАП проводилось при переполнении таймера TIM6.

Выходной сигнал на выходе PA5 регистрировался с помощью осциллографа RIGOL (рис. 12). Максимальное значение выходного сигнала 3 В. Период изменения выходного сигнала зависит от настроек таймера.

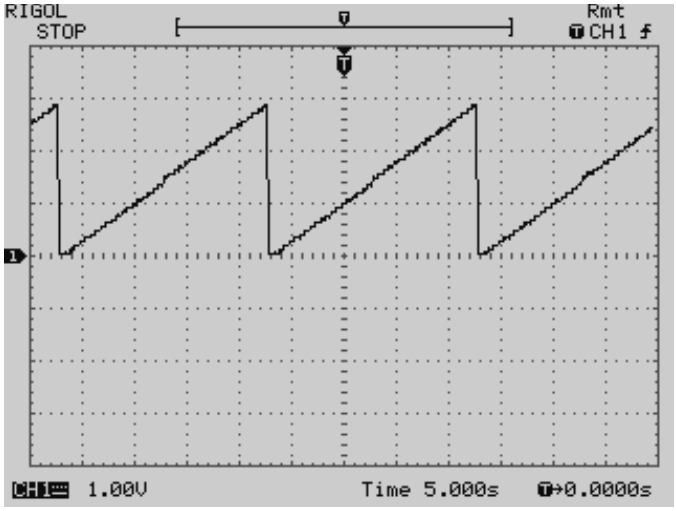

Рис. 12. Сигнал пилообразного напряжения на выходе ЦАП (PA5)

Фрагмент программного кода на языке С для формирования выходного сигнала пилообразного напряжения на выходе ЦАП приведен на рис. 13.

```
// амплитудные значения ЦАП для построения пилообразного напряжения
const uint16_t signal[40]= {
      0,  100,  200,  300,  400,  500,  600, 700,
      800, 900, 1000, 1100, 1200, 1300, 1400, 1500,
      1600, 1700, 1800, 1900, 2100, 2200, 2300, 2400,
      2500,  2600,  2700,  2800,  2900,  3000,  3100,  3200,
      3300,  3400,  3500,  3600,  3700,  3800,  3900,  4000,
       };

unsigned char i=0;

// Обработчик прерывания от таймера 6
void TIM6_DAC_IRQHandler(void) {

 TIM_ClearITPendingBit(TIM6, TIM_IT_Update); // Сбрасываем флаг
обработки прерывания от таймера

 DAC_SetChannel2Data(DAC_Align_12b_R,signal[i++]); // Записываем в
ЦАП очередной элемент массива и равняем по правому краю
  if (i==40) i=0;
}
```

Рис. 13. Формирование пилообразного напряжения на выходе ЦАП с помощью программы на языке С

Таким образом, при возникновении прерывания по переполнению таймера TIM6 в регистр ЦАП загружается следующее по порядку значение из массива для формирования выходного сигнала.

Фрагмент программного кода настройки таймера TIM6 и цифро-аналогового преобразователя DAC (второй канал) представлен на рис. 14.

```
// ***************************** ЦАП ******************************************//

      // Тактирование ЦАП
      RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);

      // Тактирование таймера TIM6
      RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6,ENABLE);

      // Настройка таймера TIM6
      TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;
      TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
      TIM_TimeBaseStructure.TIM_Prescaler     = 8400-1;
      TIM_TimeBaseStructure.TIM_Period        = 5000-1; // частота перебора
значений массива
      TIM_TimeBaseStructure.TIM_CounterMode   = TIM_CounterMode_Up;
      TIM_TimeBaseInit(TIM6, &TIM_TimeBaseStructure);
      TIM_Cmd(TIM6, ENABLE);

      TIM_ITConfig(TIM6, TIM_IT_Update, ENABLE); // включаем прерывания по
переполнению
      NVIC_EnableIRQ(TIM6_DAC_IRQn);            //Разрешение TIM6_DAC_IRQn
прерывания

      // Включаем DAC2
      DAC_Cmd(DAC_Channel_2, ENABLE);

//*****************************************************************************//
```

Рис. 14. Настройка TIM6 и DAC

Таким образом, для активизации работы таймера TIM6 и DAC подаются тактовые импульсы, как и для любого другого периферийного устройства микроконтроллера STM32F4Discovery:

RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);

RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6,ENABLE);

Далее настраивается предделитель таймера и его период (время, через которое будет возникать прерывание). Для этого используются следующие строки программного кода:

TIM_TimeBaseStructure.TIM_Prescaler = 8400-1;

TIM_TimeBaseStructure.TIM_Period = 5000-1;

Выбирается режим счета таймера (счет вверх):

TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

Разрешается прерывание по переполнению таймера TIM6:

TIM_ITConfig(TIM6, TIM_IT_Update, ENABLE);

NVIC_EnableIRQ(TIM6_DAC_IRQn);

Для функционирования DAC его необходимо включить:

DAC_Cmd(DAC_Channel_2, ENABLE);

Для программирования микроконтроллеров STM32 используется свободно распространяемая среда разработки Atollic True Studio.

Данное программное обеспечение позволяет писать, отлаживать, строить, компилировать программный код, созданный на языках программирования высокого уровня С,

C++. Оно имеет функцию подсветки синтаксиса, дополнения программного кода (т.е. среда разработки автоматически завершает написание каких-либо функций, структур, что очень удобно для разработчика). Atollic True Studio имеет дружественный интерфейс, среда очень удобна и проста в использовании. На рис. 15 представлено диалоговое окно выбора рабочего пространства (появляется при запуске среды разработки и предлагает выбрать папку будущего проекта). На рис. 16 представлено окно среды разработки при написании исходного кода на языке С. Интерфейс среды разработки при пошаговой отладке разработанной программы для микроконтроллера STM32F4Discovery представлен на рис. 17.

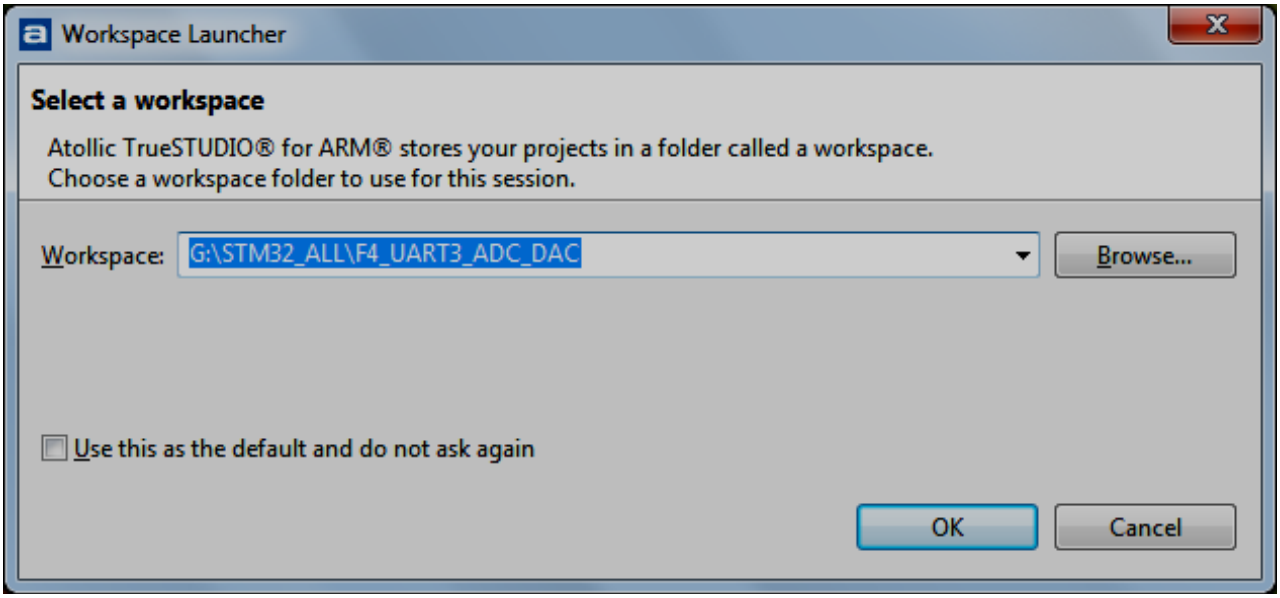

Рис. 15. Диалог выбора рабочего пространства

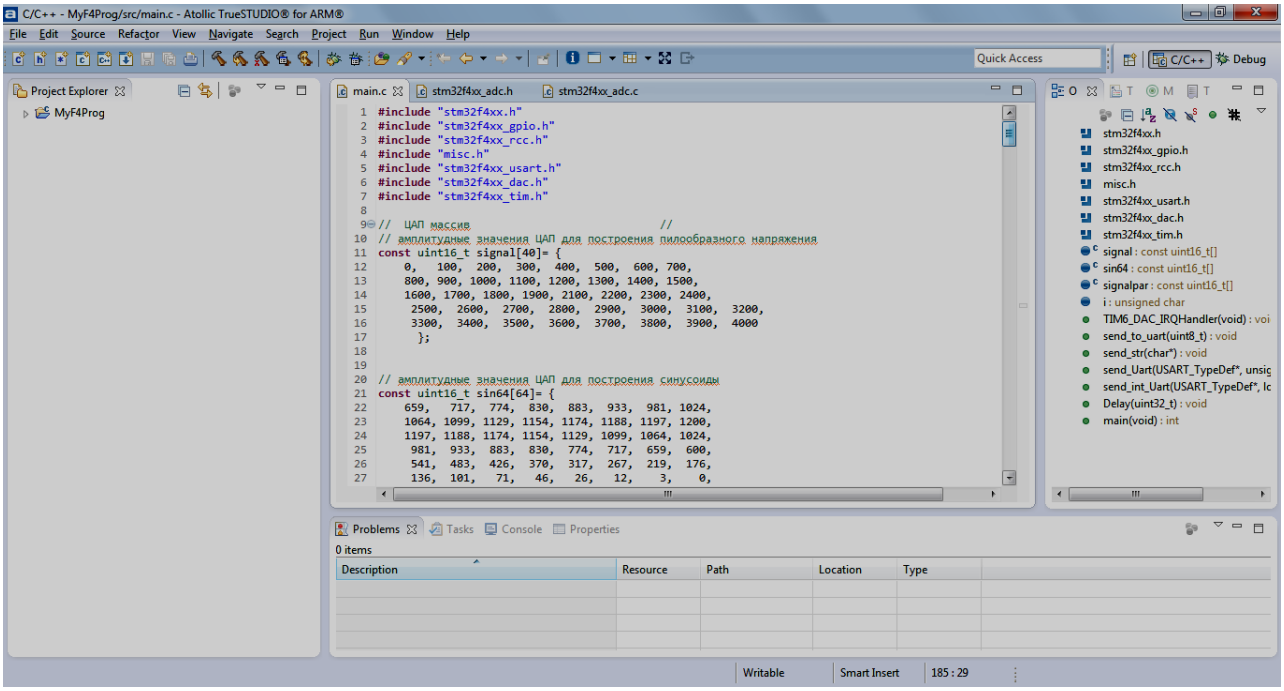

Рис. 16. Среда разработки в режиме написания программного кода

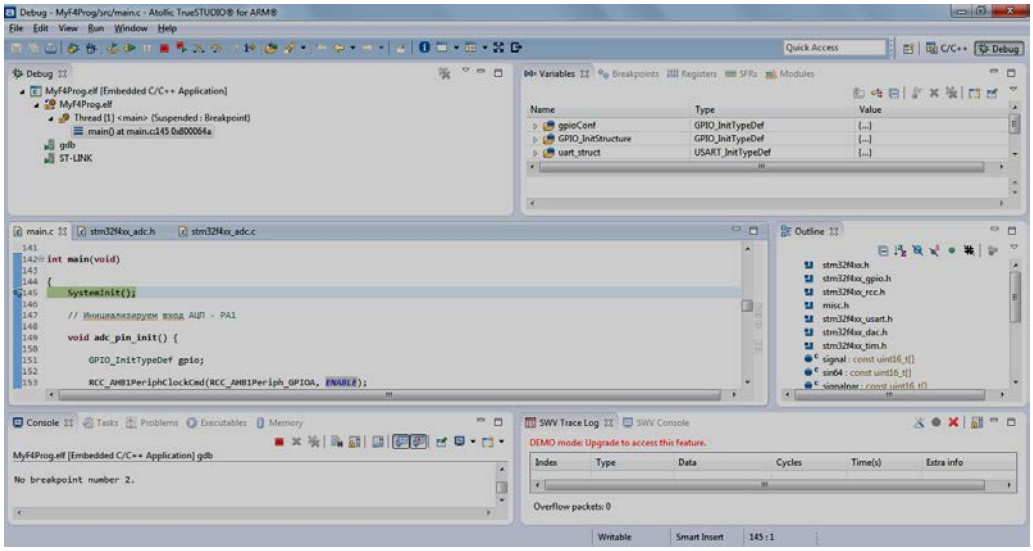

Рис.17. Интегрированная среда разработки Atollic True Studio в режиме пошаговой отладки

ВЫВОДЫ

В результате выполненной работы были исследованы возможности встроенных АЦП и ЦАП, определены особенности их настройки и программирования на языке высокого уровня С. В общем виде разработан алгоритм настройки АЦП и ЦАП микроконтроллера STM32F4Discovery. Для оценки их возможностей получены программные коды для преобразования аналогового сигнала в цифровой вид и получения выходного аналогового сигнала, сформированного с помощью загрузки цифрового массива значений ЦАП в его специализированный регистр, а также показано, как корректно настроить порты ввода-вывода и остальные периферийные устройства. Детально рассмотрен процесс настройки АЦП и ЦАП.

Таким образом, полученные результаты могут быть использованы для программирования микроконтроллеров STM32 при построении систем контроля, измерительных систем и управления.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Федорков Б.Г. Микросхемы ЦАП и АЦП: функционирование, параметры, применение / Б. Г. Федорков, В. А. Телец. – М .: Энергоатомвидат, 1990. – 320 с.

2. Быстродействующие интегральные микросхемы ЦАП и АЦП и измерение их параметров. / Под ред. А.-Й. К. Марцинкявичуса, Е.А. К. Банданскиса. – М .: Радио и связь, 1988. – 224 с.

3. Интегральные микросхемы: Микросхемы для аналого-цифрового преобразования и средств мультимедиа. – М.: ДОДЕКА, 1996. – Выпуск 1. – 384 с.

4. Нефедов А.В. Интегральные микросхемы и их зарубежные аналоги. Справочник. Т.9. – М .: ИП Радиософт, 1999. – 512 с.

5. Нефедов А.В. Интегральные микросхемы и их зарубежные аналоги. Справочник. Т.10 – М .: ИП Радиософт, 2000. – 554 с.

6. Yang H.Y., Sarpeshkar R.A. time-based energy-efficient analog-to-digital converter. IEEE Journal of Solid-State Circuits 2005. – 40 (8). – P. 1590–1601.

7. Zarifi M.H., Frounchi J., Tinati M.A., Farshchi S., Judy J.W. A low-power small-area 10-bit analog-to-digital converter for neural recording applications. International journal of circuit theory and applications Int. J. Circ. Theor. Appl. 2011. – 39. – P. 385–395.

8. Бугаев В.И.Лабораторный практикум для изучения микроконтроллеров архитектуры ARM Cortex-M4 на базе оценочного модуля STM32F4Discovery / В.И. Бугаев, М.П. Мусиенко, Я.М. Крайних. – Москва – Николаев: МФТИ.–ЧГУ, 2013. –71 с.

9. DM00031020 Reference manual. [Электронный ресурс] - Режим доступа: URL: http://www.st.com/st-web ui / static / active / en / resource / technical / document / reference_manual / DM00031020.pdf

10. DM00039084 User manual. [Электронный ресурс] - Режим доступа: URL: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00039084.pdf

Статья поступила в редакцию 18.04.2016 г.