1. Указание мер безопасности.

К работе допускаются лица, изучившие настоящее описание, инструкцию по технике безопасности при работе с измерительными приборами, а также прошедшие инструктаж по безопасности труда на рабочем месте.

Прежде чем приступить к работе, внешним осмотром оборудования убедитесь в отсутствии механических повреждений его элементов, отсутствии торчащих и оборванных проводов, следов горения электрооборудования. О выявленных недостатках сообщите преподавателю.

Приступайте к работе только после разрешения и в присутствии преподавателя.

2. Техника безопасности при измерении электрических величин

Все электроизмерительные приборы и проверяемые аппараты должны быть в устойчивом положении. Нельзя применять в качестве подставок и опор посторонние предметы (книги, инструменты и т.п.).

Перед измерением высоких напряжений и включением высоковольтных электрических установок необходимо:

- проверить наличие комплекта защитных средств (резиновых перчаток, бот, резинового коврика, переносного заземления и т.д.);
- проверить исправность системы защитных заземлений;
- надежно заземлить металлический кожух измерительного прибора (делителя напряжения), используя специальные клеммы;
- измерить сопротивление изоляции аппаратуры.

Во время измерений ***все операции выполнять только одной рукой***; вторая рука должна быть свободной (при небольшом опыте работы с электроизмерительной аппаратурой лучше всего свободную руку плотно прижать к телу).

Высоковольтный щуп, применяемый для измерения напряжений до 5 – 8 кВ держать только в резиновых перчатках. При напряжениях свыше 5 – 8 кВ применять специальные щупы на длинной заземленной штанге.

Во время измерений в цепях с напряжением выше 200 – 300 В обязательно присутствие второго лица в соответствии с правилами техники безопасности. Подключение к высоковольтным объектам измерений разрешается только при отключенном высоком напряжении. При этом следует предварительно разрядить конденсаторы высоковольтного фильтра выпрямителя путем замыкания их выводов изолированным проводником.

При измерении напряжений средних величин (до 300 В) один из щупов, если это позволяет схема исследуемой установки, соединяют с ее корпусом, а вторым щупом поочередно касаются требуемых точек цепи.

И всё же,
Будучи "Нигде",
Находишься ты
Где-то.

А.А. Мили, "По лестнице"

Лабораторная работа №5

Тема: Использование цифро-аналогового преобразователя (ЦАП) для формирования микроконтроллером заданной формы волны.

Цель: Ознакомится с архитектурой низкоуровневых библиотек и промежуточного программного обеспечения микроконтроллера. Закрепить навыки работы с осциллографом и оценочной платой MCBSTM32F200 в качестве измерительного генератора.

Постановка задачи: используя библиотеки Keil µVision5, разработать программу для микроконтроллера (МК) STM32F200, которая выдает на выходе ЦАП заданный уровень напряжения. Изучить и ввести программу, предназначенную для генерирования на выходе ЦАП микроконтроллера (МК) STM32F200 периодической волны напряжения заданной формы. Модифицировать данную программу так, чтобы она выводила сигнал с заданными амплитудными и временными характеристиками (размахом и периодом).

Материал для предварительного изучения:

1. Структура и организация библиотек программ, принципы работы с библиотеками.
2. Способы формирования аналоговых сигналов цифровыми методами.
3. Реализация цифро-аналоговых преобразователей.

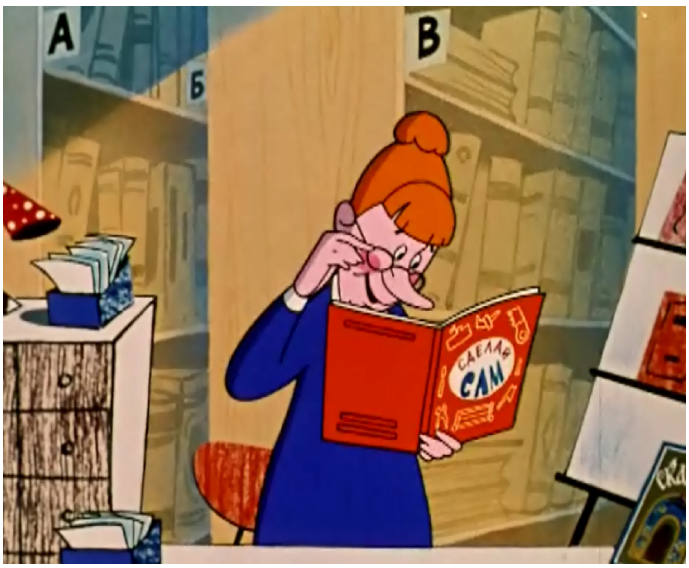
Краткие теоретические сведения

Низкоуровневые библиотеки микроконтроллера и промежуточное программное обеспечение оценочной платы MCBSTM32F200 призваны облегчить разработку программ снизить ее стоимость и уменьшить время разработки.

Стандартная периферийная библиотека STM32F2xx охватывает три абстрактных уровня и включает:

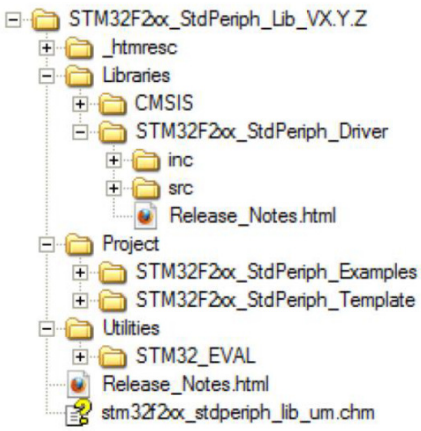
- Полную карту адресов регистров с объявленными в С всеми битами, полями битов и регистров. Это позволяет несколько упростить реализацию

задачи и, что более важно избежать ошибки вычисления адресов регистров, ускоряя начальную стадию разработки.



- Коллекцию подпрограмм и структур данных охватывающих все периферийные функции (драйверы с типичными программными интерфейсами приложений). Они могут непосредственно использоваться как структура для ссылки, поскольку они также включают макроопределения для поддержки связанных с ядром свойственных ему особенностей, общих констант и определений типов данных.
- Ряд примеров, охватывающих всю доступную периферию с шаблонами проектов для наиболее типичных разрабатываемых инструментов. С соответствующей оценочной платой, это позволяет приступить к работе с совершенно новым микроконтроллером в течение нескольких часов.

Библиотека поставляется в виде обычного заархивированного файла. Извлечение библиотеки из архива создает одну папку STM32F2xx_StdPeriph_Lib_VX.Y.Z, которая содержит следующие подпапки:



Папки содержат все CMSIS файлы и драйверы стандартной периферии микроконтроллера STM32F2xx. Более подробно с библиотекой Вы можете

познакомиться в “Description of STM32F2xx Standard Peripheral Library” (см. файл DM00023896.pdf). Дальнейшее развитие библиотеки описано в “UM1725 User Manual. Description of STM32F4xx HAL drivers[1]” (см. файл DM00105879.pdf). Мы будем использовать именно это развитие библиотеки.

В разных оценочных платах микроконтроллер STM32F2xx подключается к установленным на плате компонентам – светодиодам, кнопкам и т.п., по-разному. Для облегчения работы с компонентами, установленными на оценочной плате MCBSTM32F200, служит промежуточное программное обеспечение. Промежуточное программное обеспечение построено аналогично.

Тщательно изучайте библиотеки программ!

Программа работы

A. Проект MyDAC

1. Начните создание нового проекта. При появлении окна Manage Run-Time Environment откройте папку CMSIS и как обычно поставьте отметку в окошке CORE (поддержка ядра), в папке Device (устройство) поставьте отметку в окошке Startup (запуск), затем в папке STM32Cube Framework … отметьте окошко Classic, а в папке STM32Cube HAL отметьте окошки Common и Cortex и дополнительно DAC (ЦАП), GPIO (порты ввода-вывода общего назначения) и RCC (управление сбросом и тактированием). Если некоторые из отмеченных окошек не окрашены в зеленый цвет, нажмите кнопку Resolve в нижнем левом углу. Затем кнопку OK.
2. Завершите построение проекта созданием файла DACKeil.c. Прежде чем продолжить работу над проектом, желательно изучить ЦАП. Для этого при использовании библиотек оказывается достаточным посмотреть лишь его краткое описание в файлах CD00259245.pdf и DM00129215.pdf.
3. Теперь давайте работать вместе. Опираясь на рекомендации п. 2 2 “Overview of HAL drivers” (обзор драйверов HAL) (см. файл DM00105879.pdf) и, посмотрев на примерах, как пишутся другие программы, начнем с включения необходимых нам программ. В первой строке созданного файла наберем:

```
#include "stm32f2xx_hal.h"
```

Этот файл содержит все прототипы функций для модуля драйверов HAL (Hardware Abstraction Layer – абстрактный слой аппаратного обеспечения позволяющий управлять различными регистрами и характеристиками чипа STM32F2xx).

---

[1]Этот файл используется по той причине, что аналогичный файл для STM32F2xx пока не создан. Использование этого файла при описании периферии возможно, поскольку микроконтроллеры STM32F2xx и STM32F4xx отличаются в основном только ядром.

Хорошая культура программирования требует, чтобы затем шли частные определения типа. У нас их не будет. Теперь следует сделать собственно частные определения. Их у нас тоже нет. Сейчас очередь частных макросов. Их нет. За частными макросами идут частные переменные.

Познакомившись с п. 12 “HAL DAC Generic Driver” (общие драйверы HAL для ЦАП) (см. файл DM00105879.pdf) введем две частные переменные для ЦАП, две переменные для системы тактирования RCC и одну переменную для определения структуры инициализации портов ввода-вывода общего назначения GPIO. Первая переменная для определения структуры обработчика ЦАП, а вторая для определения постоянной структуры канала конфигурации ЦАП.

```
DAC_HandleTypeDef    DacHandle;
DAC_ChannelConfTypeDef sConfig;
```

Третья переменная для определения структуры конфигурации тактирования шин AHB и APB системы RCC. Четвертая переменная для определения структуры конфигурации внешнего/внутреннего генератора (HSE, HSI, LSE и LSI) RCC.

```
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_OscInitTypeDef RCC_OscInitStruct;
```

Пятая переменная для определения структуры инициализации портов ввода-вывода общего назначения GPIO.

```
GPIO_InitTypeDef          GPIO_InitStruct;
```

На данном этапе следует описать прототипы частных функций. Таковых у нас нет. Наконец можно переходить к описанию собственно частных функций.

В соответствии с пунктом 12.2.2 обзора драйверов HAL (см. файл DM00105879.pdf) составим функцию конфигурирования системы тактирования. Предварительно ознакомившись с п. 45 HAL RCC Generic Driver (общие драйверы HAL для RCC) (см. файл DM00105879.pdf)

Систему тактирования сконфигурируем следующим образом:

```
/**
  *         System Clock Configuration
  *         The system Clock is configured as follow :
  *            System Clock source            = PLL (HSE)
  *            SYSCLK(Hz)                     = 120000000
  *            HCLK(Hz)                       = 120000000
  *            AHB Prescaler                  = 1
  *            APB1 Prescaler                 = 4
  *            APB2 Prescaler                 = 2
  *            HSE Frequency(Hz)              = 25000000
  *            PLL_M                          = 25
  *            PLL_N                          = 240
  *            PLL_P                          = 2
```

```
  *            PLL_Q                          = 5
  *            VDD(V)                         = 3.3
  *            Flash Latency(WS)              = 3
  */
```

Функция конфигурирования системы тактирования будет выглядеть так[2]:

```
void SystemClock_Config(void)
{
   /* Enable HSE Oscillator and activate PLL with HSE as source
*/
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 25;
  RCC_OscInitStruct.PLL.PLLN = 240;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 5;
  HAL_RCC_OscConfig(& RCC_OscInitStruct);

  /* Select PLL as system clock source and configure the HCLK,
PCLK1 and PCLK2
     clocks dividers */
  RCC_ClkInitStruct.ClockType     =     (RCC_CLOCKTYPE_SYSCLK     |
RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
  HAL_RCC_ClockConfig(& RCC_ClkInitStruct, FLASH_LATENCY_3);
}
```

Теперь в соответствии с п. 12.2.2 обзора драйверов HAL (см. файл DM00105879.pdf) нужно составить функцию конфигурирования ЦАП. Сконфигурировать (включить тактирование) соответствующий канал ЦАП, и сконфигурировать выход ЦАП так, чтобы он работал в аналоговом режиме.

Описание (комментарий) конфигурирования ЦАП будет выглядеть так:

```
/**
  *        DAC initialization
  *        This function configures the hardware resources:
  *           - Peripheral's clock enable
  *           - Peripheral's GPIO Configuration
  * @param hdac: DAC handle pointer
  */
```

[2]Если Вы почувствовали, что теряете нить восприятия, самое время обратиться к справочному руководству (см. файл CD00225773.pdf). Не переживайте, понимание придёт! Здесь работает философский принцип перехода количества в качество. Помните, что в программировании понимание это всего лишь дело привычки. ☺

Функция конфигурирования ЦАП следующая:

```
void HAL_DAC_MspInit(DAC_HandleTypeDef *hdac)
{
    /*##-1-     Enable     peripherals     and     GPIO     Clocks
#################################*/
  /* Enable GPIO clock ****************************************/
  __GPIOA_CLK_ENABLE();
  /* DAC Periph clock enable */
  __DAC_CLK_ENABLE();

  /*##-2-           Configure           peripheral           GPIO
##########################################*/
  /* DAC Channel1 GPIO pin configuration */
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, & GPIO_InitStruct);
}
```

Ну и наконец, последняя функция – основное тело программы. В соответствии с п. 12.2.2 обзора драйверов HAL (см. файл DM00105879.pdf) нужно подключить канал ЦАП. Это можно сделать только после инициализации библиотеки HAL инструкцией HAL_Init(). Затем следует сконфигурировать *системное* тактирование, сконфигурировать периферию ЦАП, сконфигурировать регистры соответствующего канала ЦАП и запустить его. Все это будет выглядеть следующим образом:

```
/**
  *         Main program
  */
int main(void)
{
  /* STM32F2xx HAL library initialization:
       -  Configure  the  Flash  prefetch,  instruction  and  Data
caches
       - Configure the Systick to generate an interrupt each 1
msec
       - Set NVIC Group Priority to 4
       - Global MSP (MCU Support Package) initialization
     */
  HAL_Init();

  /* Configure the system clock to have a system clock = 120 MHz
*/
  SystemClock_Config();

  /*##-1-        Configure        the        DAC        peripheral
#####################################*/
  DacHandle.Instance = DAC;

  HAL_DAC_Init(& DacHandle);
```

```
       /*##-2-             Configure             DAC             channel1
##########################################*/
       sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
       sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_DISABLE;

       HAL_DAC_ConfigChannel(& DacHandle, & sConfig, DAC_CHANNEL_1);

       /*##-3-      Set      DAC      Channel1      DHR      register
#####################################*/
       HAL_DAC_SetValue(&  DacHandle,  DAC_CHANNEL_1,  DAC_ALIGN_12B_R,
4095);      //Ввод количества уровней квантования (например 4095)

       /*##-4-              Enable              DAC              Channel1
################################################*/
        HAL_DAC_Start(& DacHandle, DAC_CHANNEL_1);

       /* Infinite loop */
       while (1)
       {
       }
     }

     /**********************************       END       OF       FILE
**********************************/
```

4. Постройте проект, нажав F7 или выбрав пункт Build из меню Project.
5. Запустите отладочный режим, выбрав из меню Debug пункт Start/Stop Debug Session.
6. В появившемся окошке Evaluation Mode нажмите кнопку OK.
7. Подключите щуп осциллографа к гнезду PA4 площадки экспериментирования оценочной платы MCBSTM32F200. Находясь в режиме отладки, выберите из меню Debug пункт Run.
8. Программа загрузится в оценочную плату, и линия горизонтальной развертки осциллографа установится на уровне напряжения в 3 В.
9. Измените в программе значение количества уровней квантования так, чтобы ЦАП выдавал заданное преподавателем значение напряжения. Продемонстрируйте результат преподавателю.

B. Проект WaveGen

1. Начните создание нового проекта. При появлении окна Manage Run-Time Environment откройте папку CMSIS и как обычно поставьте отметку в окошке CORE (поддержка ядра), в папке Device (устройство) поставьте отметку в окошке Startup (запуск), затем в папке STM32Cube Framework … отметьте окошко Classic, а в папке STM32Cube HAL отметьте окошки Common и Cortex и дополнительно DAC (ЦАП), DMA (прямой доступ к

памяти), GPIO (порты ввода-вывода общего назначения), RCC (управление сбросом и тактированием), .TIM (таймеры). Если некоторые из отмеченных окошек не окрашены в зеленый цвет, нажмите кнопку Resolve в нижнем левом углу. Затем кнопку OK.

2. Завершите построение проекта созданием файла DACWave.c.
3. Введите программу генерирования периодической волны напряжения заданной формы. Текст программы представлен ниже:

```
/**
******************************************************************************
*       *Name:    DACWave.c
*       *Purpose: DAC for MCBSTM32F200
*
******************************************************************************
*       */

/* Includes ------------------------------------------------------------*/
#include "stm32f2xx_hal.h"

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Definition for DAC clock resources */
#define DACx_CHANNEL1_GPIO_CLK_ENABLE()  __GPIOA_CLK_ENABLE()
#define DMAx_CLK_ENABLE()                __DMA1_CLK_ENABLE()

/* Definition for DACx Channel1 Pin */
#define DACx_CHANNEL1_PIN                GPIO_PIN_4
#define DACx_CHANNEL1_GPIO_PORT          GPIOA

/* Definition for DACx's Channel1 */
#define DACx_CHANNEL1                    DAC_CHANNEL_1

/* Definition for DACx's DMA Channel1 */
#define DACx_DMA_CHANNEL1                DMA_CHANNEL_7
#define DACx_DMA_STREAM1                 DMA1_Stream5

/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
DAC_HandleTypeDef    DacHandle;
static DAC_ChannelConfTypeDef sConfig;
const uint8_t Wave[6] = {0x0, 0x33, 0x66, 0x99, 0xCC, 0xFF};
```

```
     /* Private function prototypes -----------------------------------
-------------*/
     static void DAC_Ch1_WaveConfig(void);
     static void TIM6_Config(void);
     static void SystemClock_Config(void);

     /* Private functions ---------------------------------------------
-------------*/
     /**
       * @brief DAC MSP De-Initialization
       *        This function frees the hardware resources:
       *          - Disable the Peripheral's clock
       *          - Revert GPIO to their default state
       * @param hadc: DAC handle pointer
       * @retval None
       */
     void HAL_DAC_MspInit(DAC_HandleTypeDef* hdac)
     {
       GPIO_InitTypeDef          GPIO_InitStruct;
       static DMA_HandleTypeDef  hdma_dac1;

       /*##-1- Enable peripherals and GPIO Clocks
#################################*/
       /* DAC Periph clock enable */
       __DAC_CLK_ENABLE();

       /* Enable GPIO clock ****************************************/
       DACx_CHANNEL1_GPIO_CLK_ENABLE();

       /* DMA1 clock enable */
       DMAx_CLK_ENABLE();

       /*##-2- Configure peripheral GPIO
########################################*/
       /* DAC Channel1 GPIO pin configuration */
       GPIO_InitStruct.Pin = DACx_CHANNEL1_PIN;
       GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
       GPIO_InitStruct.Pull = GPIO_NOPULL;
       HAL_GPIO_Init(DACx_CHANNEL1_GPIO_PORT, & GPIO_InitStruct);

       /*##-3- Configure the DMA streams
########################################*/
       /* Set the parameters to be configured for Channel1*/
       hdma_dac1.Instance = DACx_DMA_STREAM1;

       hdma_dac1.Init.Channel  = DACx_DMA_CHANNEL1;
       hdma_dac1.Init.Direction = DMA_MEMORY_TO_PERIPH;
       hdma_dac1.Init.PeriphInc = DMA_PINC_DISABLE;
       hdma_dac1.Init.MemInc = DMA_MINC_ENABLE;
       hdma_dac1.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
       hdma_dac1.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
       hdma_dac1.Init.Mode = DMA_CIRCULAR;
       hdma_dac1.Init.Priority = DMA_PRIORITY_HIGH;
```

```
       hdma_dac1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
       hdma_dac1.Init.FIFOThreshold = DMA_FIFO_THRESHOLD_HALFFULL;
       hdma_dac1.Init.MemBurst = DMA_MBURST_SINGLE;
       hdma_dac1.Init.PeriphBurst = DMA_PBURST_SINGLE;

       HAL_DMA_Init(& hdma_dac1);

       /* Associate the initialized DMA handle to the the DAC handle
*/
       __HAL_LINKDMA(hdac, DMA_Handle1, hdma_dac1);
     }

     /**
       * @brief TIM MSP Initialization
       *        This function configures the hardware resources:
       *           - Peripheral's clock enable
       *           - Peripheral's GPIO Configuration
       * @param htim: TIM handle pointer
       * @retval None
       */
     void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim)
     {
       /* TIM6 Periph clock enable */
       __TIM6_CLK_ENABLE();
     }

     /**
       * @brief  Main program.
       * @param  None
       * @retval None
       */
     int main(void)
     {
       /* STM32F2xx HAL library initialization:
            - Configure the Flash prefetch, instruction and Data
caches
            - Configure the Systick to generate an interrupt each 1
msec
            - Set NVIC Group Priority to 4
            - Global MSP (MCU Support Package) initialization
          */
       HAL_Init();
       /* Configure the system clock to have a system clock = 120 MHz
*/
       SystemClock_Config();

       /*##-1- Configure the DAC peripheral
######################################*/
       DacHandle.Instance = DAC;

       /*##-2- Configure the TIM peripheral
######################################*/
       TIM6_Config();
```

```
          /* Wave generator ----------------------------------------
*/
             DAC_Ch1_WaveConfig();

       /* Infinite loop */
       while (1) {}
     }

     /**
       * @brief  System Clock Configuration
       *         The system Clock is configured as follow :
       *            System Clock source            = PLL (HSE)
       *            SYSCLK(Hz)                     = 120000000
       *            HCLK(Hz)                       = 120000000
       *            AHB Prescaler                  = 1
       *            APB1 Prescaler                 = 4
       *            APB2 Prescaler                 = 2
       *            HSE Frequency(Hz)              = 25000000
       *            PLL_M                          = 25
       *            PLL_N                          = 240
       *            PLL_P                          = 2
       *            PLL_Q                          = 5
       *            VDD(V)                         = 3.3
       *            Flash Latency(WS)              = 3
       * @param  None
       * @retval None
       */
     static void SystemClock_Config(void)
     {
       RCC_ClkInitTypeDef RCC_ClkInitStruct;
       RCC_OscInitTypeDef RCC_OscInitStruct;

       /* Enable HSE Oscillator and activate PLL with HSE as source */
       RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
       RCC_OscInitStruct.HSEState = RCC_HSE_ON;
       RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
       RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
       RCC_OscInitStruct.PLL.PLLM = 25;
       RCC_OscInitStruct.PLL.PLLN = 240;
       RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
       RCC_OscInitStruct.PLL.PLLQ = 5;
       HAL_RCC_OscConfig(& RCC_OscInitStruct);

       /* Select PLL as system clock source and configure the HCLK,
PCLK1 and PCLK2
          clocks dividers */
       RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK |
RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
       RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
       RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
       RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
       RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
```

```
    HAL_RCC_ClockConfig(& RCC_ClkInitStruct, FLASH_LATENCY_3);
  }

  static void DAC_Ch1_WaveConfig(void)
  {
    /*##-1- Initialize the DAC peripheral
######################################*/
    HAL_DAC_Init(& DacHandle);

    /*##-2- DAC channel1 Configuration
#########################################*/
    sConfig.DAC_Trigger = DAC_TRIGGER_T6_TRGO;
    sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;

    HAL_DAC_ConfigChannel(& DacHandle, & sConfig, DACx_CHANNEL1);

    /*##-3- Enable DAC Channel1 and associeted DMA
#############################*/
    HAL_DAC_Start_DMA(& DacHandle, DACx_CHANNEL1, (uint32_t *)Wave,
6, DAC_ALIGN_12B_R);

    /*##-4- Enable DAC Channel1
################################################*/
    HAL_DAC_Start(& DacHandle, DACx_CHANNEL1);

    /*##-5- Set DAC channel1 DHR12RD register
############################################*/
    HAL_DAC_SetValue(& DacHandle, DACx_CHANNEL1, DAC_ALIGN_12B_R,
0x100);
  }

  /**
    * @brief  TIM6 Configuration
    * @note   TIM6 configuration is based on APB1 frequency
    * @note   TIM6 Update event occurs each TIM6CLK/256
    * @param  None
    * @retval None
    */
  void TIM6_Config(void)
  {
    static TIM_HandleTypeDef  htim;
    TIM_MasterConfigTypeDef MasterConfig;

    /*##-1- Configure the TIM peripheral
######################################*/
    /* Time base configuration */
    htim.Instance = TIM6;

    htim.Init.Period = 0x7FF;
    htim.Init.Prescaler = 0;
    htim.Init.ClockDivision = 0;
    htim.Init.CounterMode = TIM_COUNTERMODE_UP;
    HAL_TIM_Base_Init(& htim);
```

```
        /* TIM6 TRGO selection */
        MasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
        MasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;

        HAL_TIMEx_MasterConfigSynchronization(& htim, & MasterConfig);

        /*##-2- Enable TIM peripheral counter
##################################*/
        HAL_TIM_Base_Start(& htim);
    }

    /************************************ END OF FILE
*****************************/
```

4. Постройте проект, нажав F7 или выбрав пункт Build из меню Project.
5. Запустите отладочный режим, выбрав из меню Debug пункт Start/Stop Debug Session.
6. В появившемся окошке Evaluation Mode нажмите кнопку OK.
7. Подключите щуп осциллографа к гнезду PA4 площадки экспериментирования оценочной платы MCBSTM32F200. Находясь в режиме отладки, выберите из меню Debug пункт Run.
8. Программа загрузится в оценочную плату, и на экране осциллографа, после его синхронизации нажатием на кнопку “Autoset”, появится периодический ступенчатый сигнал из 6 ступенек.
9. Используя осциллограф, измерьте размах напряжения и период повторения сигнала.
10. Модифицируйте данную программу так, чтобы она выводила синусоидальный сигнал с заданными амплитудными и временными характеристиками (размахом и периодом).
11. Используя осциллограф, измерьте размах и амплитуду напряжения синусоидального сигнала, период его повторения и его спектральный состав.

ВЫПОЛНЕНИЕ МАТЕМАТИЧЕСКИХ ОПЕРАЦИЙ НА ВХОДНЫХ КАНАЛАХ

ЦЕЛИ

В конце этой лабораторной сессии Вы будете способны:

- Захватывать и показывать сигнал от заданного испытуемого прибора.
- Использовать математические функции осциллографа для:
  - Выполнения передовых операций, таких как быстрое преобразование Фурье входных сигналов в каналах

ОБОРУДОВАНИЕ

Для выполнения этого эксперимента Вам потребуется:

- Испытуемый прибор в качестве источника сигнала, например, оценочная плата MCBSTM32F200 или эквивалентный генератор сигнала
- Осциллограф (TBS 1202B - EDU)
- Пассивный пробник напряжения с ослаблением 10X (TPP0101 или P5050) и BNC кабель

ТЕОРИЯ

Для выполнения этого эксперимента нам потребуется познакомиться с:

- Инструкцией пользователя осциллографом
- XYZ осциллографов – страница 32, раздел по математическим операциям
- Базовыми концепциями быстрого преобразования Фурье (БПФ) сигналов.
- TBS инструкцией пользователя осциллографом – страница 51 БПФ
- Преобразованием Фурье/представлением сигнала в частотной области

Ключевые концепции:

- Быстрое преобразование Фурье или кратко БПФ является алгоритмом вычисления дискретного преобразования Фурье временных рядов/сигналов наиболее быстрым способом.
- БПФ используется для представления изменяющегося во времени сигнала в частотной области. Любой сигнал временной области может быть представлен в частотной области комбинацией основной частоты и ее гармоник. БПФ помогает нам разделить и визуализировать сигнал временной области на его частотные составляющие.
- N-точечное БПФ сигнала, дискретизация которого выполнена со скоростью f_S отсчетов в секунду, в результате даст составляющие с частотами от 0 Гц до $f_S/2$ Гц с разрешением по частоте равным f_S/N Гц.
- Чистая синусоидальная волна в БПФ спектре даст одну частотную составляющую.

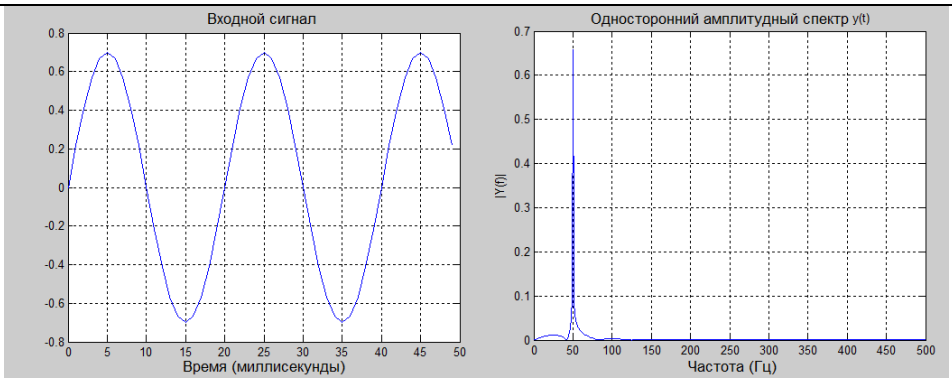


Рисунок 1: Синусоидальная форма волны/представление сигнала во временной и частотной областях

- Любая сложная волна, например прямоугольная волна, кроме основной частоты будет иметь кратные составляющие частоты (называемые гармониками).

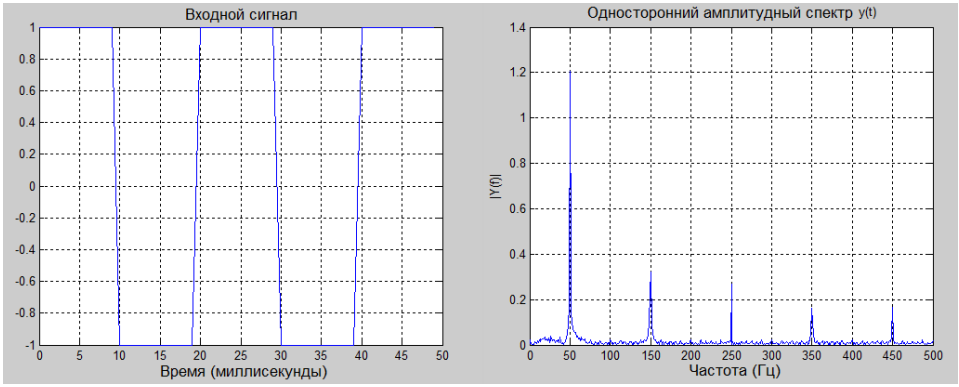


Рисунок 2: Прямоугольная форма волны/представление сигнала во временной и частотной областях

- В вычислении БПФ важную роль играет теорема В.А. Котельникова, доказанная им в 1933 году (в зарубежной литературе – теорема Найквиста). Любой изменяющийся во времени сигнал, обладающий максимальной частотной составляющей f_m, может быть полностью восстановлен, если его дискретизация была выполнена со скоростью большей, чем $2f_m$. Дискретизация с меньшей скоростью называется недодискретизацией и приводит к подмене имен – явлению, при котором высокочастотные составляющие кажутся (ложно) более низкочастотными.

ПРОВЕРЬТЕ СВОЕ ПОНИМАНИЕ

Ответьте на следующее:

1. Спектр БПФ синусоидальной волны с периодом 20 микросекунд будет обладать частотными составляющими с частотами:
   [A] 5 кГц, 15 кГц, 25 кГц …
   [B] Только 5 кГц
   [C] 50 кГц, 150 кГц, 250 кГц …
   [D] Только 50 кГц

2. Спектр БПФ прямоугольной волны с периодом 20 микросекунд будет обладать частотными составляющими с частотами:
   [A] 5 кГц, 15 кГц, 25 кГц …
   [B] Только 5 кГц
   [C] 50 кГц, 150 кГц, 250 кГц …
   [D] Только 50 кГц

ПОДГОТОВКА К ЭКСПЕРЕМЕНТИРОВАНИЮ

Шаг 1: **Подготовка испытуемого прибора**	
Шаг 2: **Подготовка измерения**	

ЛАБОРАТОРНЫЙ ОПЫТ

Для заданного сигнала:	**Ссылка на идентификатор формы волны:**
1. С помощью математических функций оцените БПФ заданного сигнала **2.** Для различных окон, таких как Хенинга, с плоской вершиной в частотной области и прямоугольное, оцените разницу спектров способа ограничения входного сигнала окном (используйте для вычислений БПФ). **3.** Если потребуется, используйте для рассматривания спектра масштабирование частоты	

Процедура/Алгоритм

Установки оценочной платы MCBSTM32F200		
	Канал № 1	Канал № 2
➢ Тип сигнала	Синусоидальная волна	
➢ Размах	2,4 – 3,3 вольта (от пика до пика)	
➢ Частота	1 кГц	

Блок-схема процедуры:

Наблюдения (Выполнение)
Мы наблюдаем, что: ➢ ➢ ➢ **Скриншоты:**
В этом эксперименте мы изучили: ➢ ➢ ➢
Данные концепции мы можем использовать в приложениях реальной жизни: ➢ ➢ ➢

ПОСТЛАБОРАТОРНОЕ ОЦЕНИВАНИЕ

Ответьте на следующее:

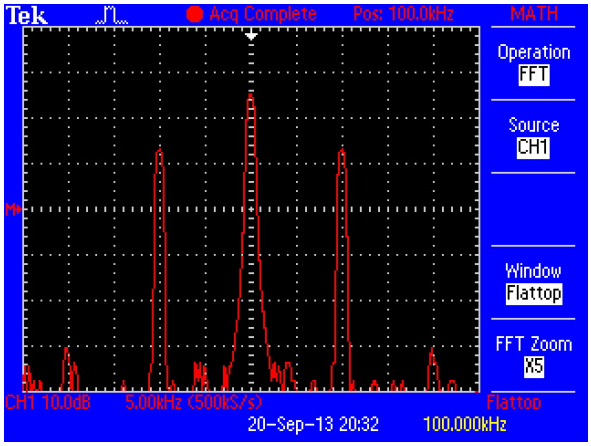
1. Перечислите концепции, которые Вы изучили, выполняя это упражнение.

2. Перечислите 8 экспериментов предписанных учебным планом, где Вы можете применить концепции измерений изученные в данном эксперименте? Какие параметры Вы будете измерять в тех экспериментах?

№ п/п	Название экспериментов предписанных учебным планом	Измеряемый параметр
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		

3. На следующем скриншоте представлен спектр БПФ амплитудно-модулированного (АМ) сигнала. Можете ли Вы выяснить глядя на спектр каковы частоты модулирующего колебания (f_M) и несущей (f_C) (мы видим 3 пика при – 90 кГц, 100 кГц и 110 кГц соответственно)? Амплитудная модуляция является результатом перемножения колебаний разных частот. Чтобы прийти к правильному ответу перемножьте два синуса разных частот.

[A] f_M = 100 кГц и f_C = 90 кГц
[B] f_M = 100 кГц и f_C = 110 кГц
[C] f_M = 100 кГц и f_C = 10 кГц
[D] f_M = 10 кГц и f_C = 100 кГц



12. Используя осциллограф, продемонстрируйте полученный результат преподавателю.

Содержание отчета:

1. Тема и цель работы.
2. Задание на лабораторную работу.
3. Алгоритм программы.
4. Полученные результаты.
5. Выводы по работе с анализом реализованной программы.

Литература:

1. UM1061 Description of STM32F2xx Standard Peripheral Library. Доступно (2015, Июнь): http://www.st.com/

2. AN3126 Application note. Audio and waveform generation using the DAC in STM32 microcontrollers. Доступно (2015, Июнь): http://www.st.com/

3. AN4566 Application note. Extending the DAC performance of STM32 microcontrollers. Доступно (2015, Июнь): http://www.st.com/