МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»

Институт компьютерных наук и технологий

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

Отчет о выполнении лабораторной работы №5 «Использование цифро-аналогового преобразователя (ЦАП) для формирования микроконтроллером заданной формы волны.» Курс «Программирование микроконтроллеров»

Выполнил студент группы

№5130201/30001 _____ Мелещенко С. И.

Преподаватель _____ Вербова Н. М.

«_____»_____ 2025г.

Санкт-Петербург, 2025

Введение

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 Vision5, 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).

1 Краткие теоритические 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 C 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_StdPerip_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”. Дальнейшее развитие библиотеки описано в “UM1725 User Manual. Description of STM32F4xx HAL drivers ”. 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.

2 Основная часть

2.1 Алгоритм программы

1. Конфигурация системы Программа начинается с инициализации HAL-библиотек микроконтроллера. Затем выполняется настройка системного тактирования, где задаются параметры генерации тактовых сигналов, включая частоту процессора и периферии.

2. Инициализация периферийных устройств

Включение тактирования ЦАП и GPIO, настройка GPIO в аналоговом режиме для работы с ЦАП и создание переменных для конфигурации ЦАП и тактирования.

3. Настройка и запуск ЦАП

Выбор режима работы без триггера, отключение выходного буфера, установка значения на выходе ЦАП, где 12-разрядное число определяет уровень выходного напряжения, запуск ЦАП.

4. Генерация периодического сигнала

Используется второй проект (WaveGen), где дополнительно задействуется прямой доступ к памяти (DMA) и таймер (TIM6). Генерация сигналов происходит путем передачи массива значений в ЦАП через DMA. Таймер TIM6 управляет частотой выборки данных из массива, задавая период выходного сигнала.

5. Выполнение математических операций

Производится быстрое преобразование Фурье входного сигнала с использованием осциллографа, анализируются частотные составляющие сигнала, измеряются размах и период волны.

2.2 Код программы

Общий код программы приведен в приложении.

1. Подключение заголовочных файлов

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

Этот заголовочный файл содержит определения всех функций и структур, необходимых для работы с библиотекой HAL (Hardware Abstraction Layer). Он включает:

Управление тактированием, GPIO, ЦАП, DMA и другими периферийными устройствами.

Определения структур данных для конфигурации периферии.

Макросы для удобного доступа к регистрам микроконтроллера.

2. Определение макросов (константы и настройки)

```
#define DACx_CHANNEL1_GPIO_CLK_ENABLE()   __GPIOA_CLK_ENABLE()
#define DMAx_CLK_ENABLE()                 __DMA1_CLK_ENABLE()
#define DACx_CHANNEL1_PIN                 GPIO_PIN_4
#define DACx_CHANNEL1_GPIO_PORT           GPIOA
#define DACx_CHANNEL1                     DAC_CHANNEL_1
#define DACx_DMA_CHANNEL1                 DMA_CHANNEL_7
#define DACx_DMA_STREAM1                  DMA1_Stream5
```

Эти макросы предназначены для удобства работы с периферией:

DACx_CHANNEL1_GPIO_CLK$_E NABLE() : GPIOA$.

DMAx_CLK_ENABLE(): Включает тактирование контроллера DMA1.

DACx_CHANNEL1_PIN: Определяет номер вывода микроконтроллера, к которому подключен выход ЦАП.

DACx_CHANNEL1_GPIO_PORT: Указывает порт (GPIOA), к которому подключен выход ЦАП.

DACx_CHANNEL1: Указывает, что используется канал 1 ЦАП.

DACx_DMA_CHANNEL1: Определяет канал DMA для работы с ЦАП.

DACx_DMA_STREAM1: Определяет поток DMA1, связанный с ЦАП.

3. Объявление глобальных переменных

```
DAC_HandleTypeDef     DacHandle;
static DAC_ChannelConfTypeDef sConfig;
const uint8_t Wave[12] = {0x0, 0x33, 0x66, 0x99, 0xCC,
0xFF,0xFF,0xCC,0x99,0x66,0x33,0x0};
```

DacHandle: Дескриптор, хранящий настройки для работы с ЦАП.

sConfig: Структура для конфигурации канала ЦАП.

Wave[12]: Массив, содержащий данные для генерации ступенчатого сигнала (12 значений).

4. Прототипы функций

```
static void DAC_Ch1_WaveConfig(void);
static void TIM6_Config(void);
static void SystemClock_Config(void);
```

DAC_Ch1_WaveConfig(): Настраивает и запускает генерацию сигнала через ЦАП.

TIM6_Config(): Настраивает таймер TIM6 для управления ЦАП.

SystemClock_Config(): Настраивает системное тактирование.

5. Функция инициализации ЦАП и DMA

```
void HAL_DAC_MspInit(DAC_HandleTypeDef* hdac)
{
  GPIO_InitTypeDef           GPIO_InitStruct;
  static DMA_HandleTypeDef  hdma_dac1;

  // Включаем тактирование периферии
  __DAC_CLK_ENABLE();
  DACx_CHANNEL1_GPIO_CLK_ENABLE();
  DMAx_CLK_ENABLE();

  // Настраиваем GPIO для выхода ЦАП
  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);

  // Настраиваем DMA
  hdma_dac1.Instance = DACx_DMA_STREAM1;
  hdma_dac1.Init.Channel  = DACx_DMA_CHANNEL1;
  hdma_dac1.Init.Direction = DMA_MEMORY_TO_PERIPH; // Передача
данных из памяти в периферию
  hdma_dac1.Init.PeriphInc = DMA_PINC_DISABLE; // Инкремент
адреса периферии отключен
  hdma_dac1.Init.MemInc = DMA_MINC_ENABLE; // Инкремент адреса
памяти включен
  hdma_dac1.Init.PeriphDataAlignment = DMA_PDATAALIGN_BYTE;
  hdma_dac1.Init.MemDataAlignment = DMA_MDATAALIGN_BYTE;
  hdma_dac1.Init.Mode = DMA_CIRCULAR; // Круговой режим
  hdma_dac1.Init.Priority = DMA_PRIORITY_HIGH;
  hdma_dac1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;

  HAL_DMA_Init(&hdma_dac1);
  __HAL_LINKDMA(hdac, DMA_Handle1, hdma_dac1);
}
```

Эта функция:
Настраивает порт GPIO в аналоговом режиме.
Включает тактирование ЦАП и DMA.
Конфигурирует DMA для передачи данных из памяти в ЦАП.
6. Инициализация таймера TIM6

```
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim)
{
  __TIM6_CLK_ENABLE();
}
```

Функция включает тактирование таймера TIM6.
7. Главная функция (основной цикл)

```
int main(void)
{
  HAL_Init(); // Инициализация HAL
  SystemClock_Config(); // Настройка системного тактирования

  DacHandle.Instance = DAC; // Указываем, что работаем с ЦАП
  TIM6_Config(); // Настраиваем таймер
  DAC_Ch1_WaveConfig(); // Настраиваем и запускаем генерацию
сигнала

  while (1) {} // Бесконечный цикл
}
```

Инициализирует HAL и системное тактирование.
Настраивает таймер и ЦАП.
Запускает генерацию сигнала.
Работает в бесконечном цикле.
8. Конфигурация тактирования

```
static void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_OscInitTypeDef RCC_OscInitStruct;

  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);

  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);
}
```

Эта функция настраивает тактирование микроконтроллера с использованием PLL.

9. Конфигурация и запуск ЦАП

```
static void DAC_Ch1_WaveConfig(void)
{
  HAL_DAC_Init(& DacHandle);
  sConfig.DAC_Trigger = DAC_TRIGGER_T6_TRGO;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;

  HAL_DAC_ConfigChannel(& DacHandle, & sConfig, DACx_CHANNEL1);
  HAL_DAC_Start_DMA(& DacHandle, DACx_CHANNEL1, (uint32_t *)
  Wave, 12, DAC_ALIGN_12B_R);
  HAL_DAC_Start(& DacHandle, DACx_CHANNEL1);
  HAL_DAC_SetValue(& DacHandle, DACx_CHANNEL1,
  DAC_ALIGN_12B_R, 0x100);
}
```

Инициализирует и запускает ЦАП.

Запускает DMA для передачи массива Wave[].

10. Настройка таймера TIM6

```
void TIM6_Config(void)
{
  static TIM_HandleTypeDef  htim;
  TIM_MasterConfigTypeDef MasterConfig;

  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);
```

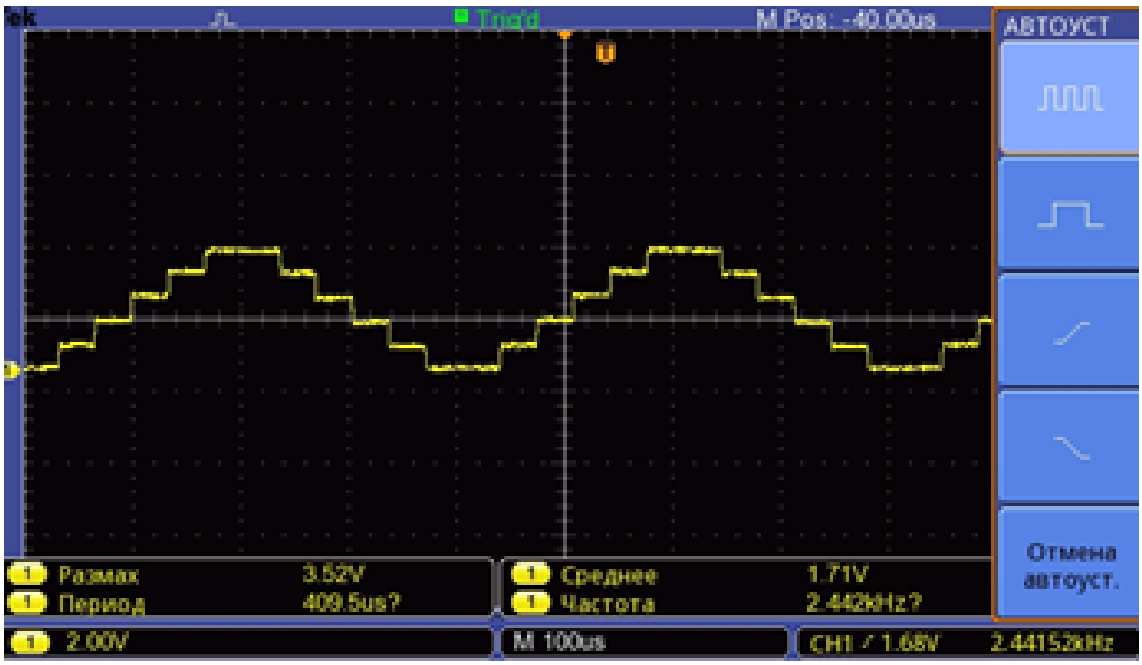
```
    MasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
    HAL_TIMEx_MasterConfigSynchronization(&htim, &MasterConfig);
    HAL_TIM_Base_Start(&htim);
}
```

Настраивает таймер для управления частотой обновления ЦАП.

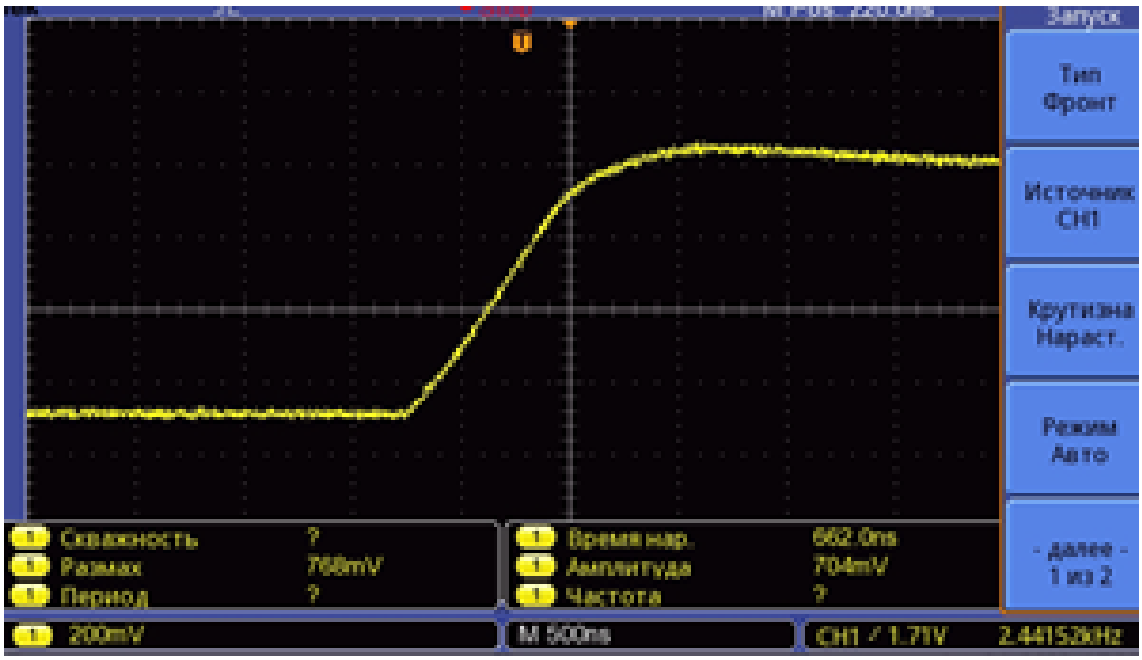
2.3 Работа с осцилографом

Прямолинейный сигнал с размахом 2.

Сигнал в форме синусоиды.



Изменение уровня сигнала



2.4 Ответы на вопросы

1. Спектр БПФ синусоидальной волны с периодом 20 микросекунд содержит частотные составляющие:

[B] Только 50 кГц.

2. Спектр БПФ прямоугольной волны с периодом 20 микросекунд содержит частотные составляющие:

[D] 5 кГц, 15 кГц, 25 кГц ....

2.5 Постлабораторное оценивание

1. Концепции, изученные в ходе работы:

Основы работы цифро-аналогового преобразования.

Использование HAL-библиотек для работы с периферией STM32.

Работа с осциллографом и анализ сигналов.

Применение быстрого преобразования Фурье для анализа частотных составляющих сигнала.

3. Частоты модулирующего и несущего колебаний в AM сигнале:

[D] fM = 10 кГц и fC = 100 кГц.

3 Вывод

В ходе выполнения лабораторной работы была изучена работа ЦАП микроконтроллера STM32F200, а также способы генерации аналоговых сигналов цифровыми методами. Освоена работа с библиотеками HAL, использован механизм DMA для автоматической передачи данных в ЦАП. Проведен спектральный анализ сигнала с помощью быстрого преобразования Фурье, что позволило выявить его частотные характеристики. Полученные знания могут быть применены в различных практических задачах, связанных с обработкой и генерацией сигналов. Программа успешно сформировала аналоговый сигнал на выходе ЦАП микроконтроллера, осциллограф подтвердил выходное напряжение 3 В, была проведена модификация кода для генерации синусоидального сигналаи проведено БПФ анализа сигнала, получены спектральные характеристики.

Приложение

Листинг 1: Код программы

```
#include "stm32f2xx_hal.h"

#define DACx_CHANNEL1_GPIO_CLK_ENABLE()   __GPIOA_CLK_ENABLE()
#define DMAx_CLK_ENABLE()                 __DMA1_CLK_ENABLE()

#define DACx_CHANNEL1_PIN                 GPIO_PIN_4
#define DACx_CHANNEL1_GPIO_PORT           GPIOA
#define DACx_CHANNEL1                     DAC_CHANNEL_1
#define DACx_DMA_CHANNEL1                 DMA_CHANNEL_7
#define DACx_DMA_STREAM1                  DMA1_Stream5
DAC_HandleTypeDef     DacHandle;
static DAC_ChannelConfTypeDef sConfig;
const uint8_t Wave[12] = {0x0, 0x33, 0x66, 0x99,
0xCC, 0xFF,0xFF,0xCC,0x99,
0x66,0x33,0x0};
static void DAC_Ch1_WaveConfig(void);
static void TIM6_Config(void);
static void SystemClock_Config(void);
void HAL_DAC_MspInit(DAC_HandleTypeDef* hdac)
{
  GPIO_InitTypeDef          GPIO_InitStruct;
  static DMA_HandleTypeDef  hdma_dac1;
  __DAC_CLK_ENABLE();
  DACx_CHANNEL1_GPIO_CLK_ENABLE();
  DMAx_CLK_ENABLE();
  GPIO_InitStruct.Pin = DACx_CHANNEL1_PIN; // Pin      номер
  GPIO_InitStruct.Mode = GPIO_MODE_ANALOG; //       аналоговый вход.
  GPIO_InitStruct.Pull = GPIO_NOPULL; // Pull      режим подтягивающего резистора.
GPIO_NOPULL      резистор отключен.
  HAL_GPIO_Init(DACx_CHANNEL1_GPIO_PORT, & GPIO_InitStruct);
  hdma_dac1.Instance = DACx_DMA_STREAM1;

  hdma_dac1.Init.Channel  = DACx_DMA_CHANNEL1;
  hdma_dac1.Init.Direction = DMA_MEMORY_TO_PERIPH; - Peripheral to memory direction.
  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);
  __HAL_LINKDMA(hdac, DMA_Handle1, hdma_dac1);
}

void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim)
{
  __TIM6_CLK_ENABLE();
}
```

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  DacHandle.Instance = DAC;
  TIM6_Config();
        DAC_Ch1_WaveConfig();
  while (1) {}
}
static void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_OscInitTypeDef RCC_OscInitStruct;
  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);
  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)
{
  HAL_DAC_Init(& DacHandle);
  sConfig.DAC_Trigger = DAC_TRIGGER_T6_TRGO;
  sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;

  HAL_DAC_ConfigChannel(& DacHandle, & sConfig, DACx_CHANNEL1);
  HAL_DAC_Start_DMA(& DacHandle, DACx_CHANNEL1, (uint32_t *)Wave, 12, DAC_ALIGN_12B_R);
  HAL_DAC_Start(& DacHandle, DACx_CHANNEL1);
  HAL_DAC_SetValue(& DacHandle, DACx_CHANNEL1, DAC_ALIGN_12B_R, 0x100);
}

void TIM6_Config(void)
{
  static TIM_HandleTypeDef  htim;
  TIM_MasterConfigTypeDef MasterConfig;
  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);

  MasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
  MasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;

  HAL_TIMEx_MasterConfigSynchronization(& htim, & MasterConfig);
  HAL_TIM_Base_Start(& htim);}
```