САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Высшая школа искусственного интеллекта
Направление 3.02.01 Математика и Компьютерные науки

Отчёт по дисциплине Программирование микроконтроллеров.

Лабораторная работа № 5.

Работу выполнил:
Путята М.А.
студент группы 5130201/30002
Проверила:
Вербова Н. М.

Санкт-Петербург - 2025 г.

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).

Теоретические данные:

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

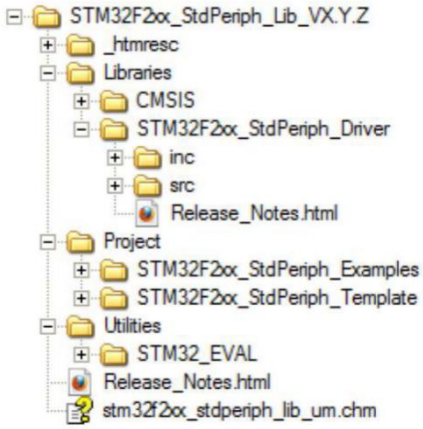
Стандартная периферийная библиотека STM32F2xx охватывает три абстрактных уровня и включает:

- 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.

Осцилля́тор — система, совершающая колебания, то есть показатели которой периодически повторяются во времени.

---

[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.

Внутренний PLL — система фазовой автоподстройки частоты (ФАПЧ) В нашем случае можно сказать, что это умножитель частоты с управляемым коэффициентом умножения.

HCLK to AHB – тактирование шины AHB. К этой шине подключены: ядро процессора, память и контроллер прямого доступа к памяти (DMA). Синхроимпульсы для нее также поступают непосредственно от сигнала HCLK . Ядро процессора и его шина (AHB) синхронизируются одним сигналом.

Direct memory access (DMA) - прямой доступ к памяти (ПДП) используется для быстрой передачи данных между памятью и периферийным устройством, памятью и памятью, или между двумя периферийными устройствами без участия процессора

Код программы:

```
#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[12] = {0x0, 0x33, 0x66, 0x99, 0xCC,
0xFF,0xFF,0xCC,0x99,0x66,0x33,0x0};

/* 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; // 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);

  /*##-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;  -  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);

  /* 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 associated DMA ##########################*/
  HAL_DAC_Start_DMA(& DacHandle, DACx_CHANNEL1, (uint32_t *)Wave, 12, 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);
}
```

Алгоритм программы:

*Большое количество комментариев находится в коде программы.

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

Далее определяем макросы и глобальные переменные. Среди них массив для формирования волнообразного сигнала (синусоиды).

Объявляем прототипы функций для АЦП и таймеров.

Часы ЦАП включаются в функции void HAL_DAC_MspInit (DAC_HandleTypeDef* hdac), в этой функции происходит краткая деинициализация DAC MSP – освобождение апаратных ресурсов:

-Отключить часы периферийного устройства

-Вернуть GPIO в состояние по умолчанию

Устанавливаем параметры, которые необходимо настроить для канала 1. И Связываем инициализированный дескриптор DMA с дескриптором DAC.

В следующей функции краткая инициализация TIM MSP. Эта функция настраивает аппаратные ресурсы:

-Часы периферийного устройства включены

-Конфигурация GPIO периферии

-указатель дескриптора TIM

В основном теле программы производится:

- Настройка предварительной выборки флэш-памяти, инструкций и кешей данных.
- Настройте Systick для генерации прерывания каждую 1 мс
- Установка приоритет группы NVIC на 4
- Глобальная инициализация MSP (пакет поддержки MCU)

Генерация волнообразного сигнала: в main вызывается функция генерации сигнала и запускается бесконечный цикл while(1). На этом тело основной функции main заканчивается.

Далее пишем функцию SystemClock_Config, в которой:

-включаем осциллятор HSE и активировать PLL с HSE в качестве источника, значения устанавливаем в соответствии с данными из методического пособия

- выбераем PLL в качестве источника системного таймера и настраиваем HCLK, PCLK1 и PCLK2 – делители часов.

Далее пишем функцию DAC_Ch1_WaveConfig, в которой:

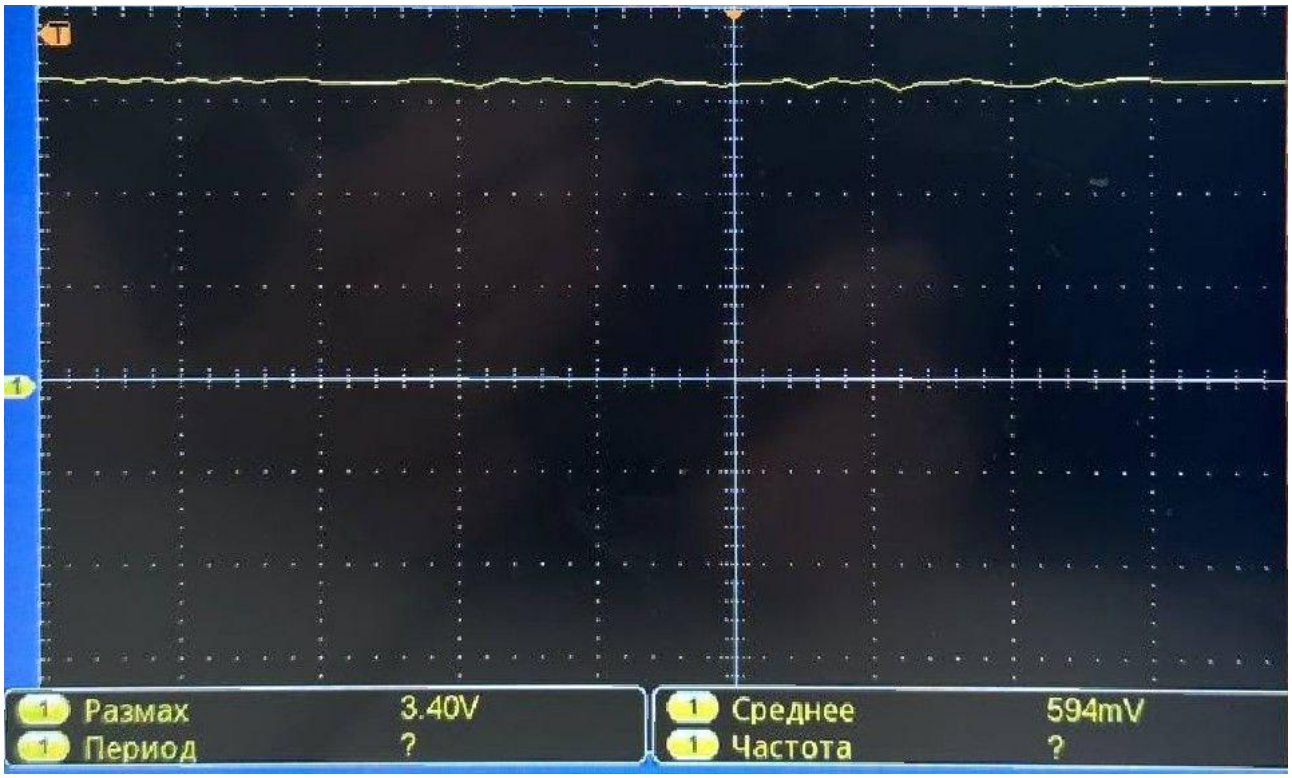
- Инициализируем периферийное устройство ЦАП
- Конфигурируем канал1 ЦАП
- Включаем DAC Channel1 и связанный с ним DMA
- Включить DAC Channel1
- Установка регистра DHR12RD канала 1 ЦАП

И наконец переопределяем функцию TIM6_Config, в которой происходит конфигурация TIM6:

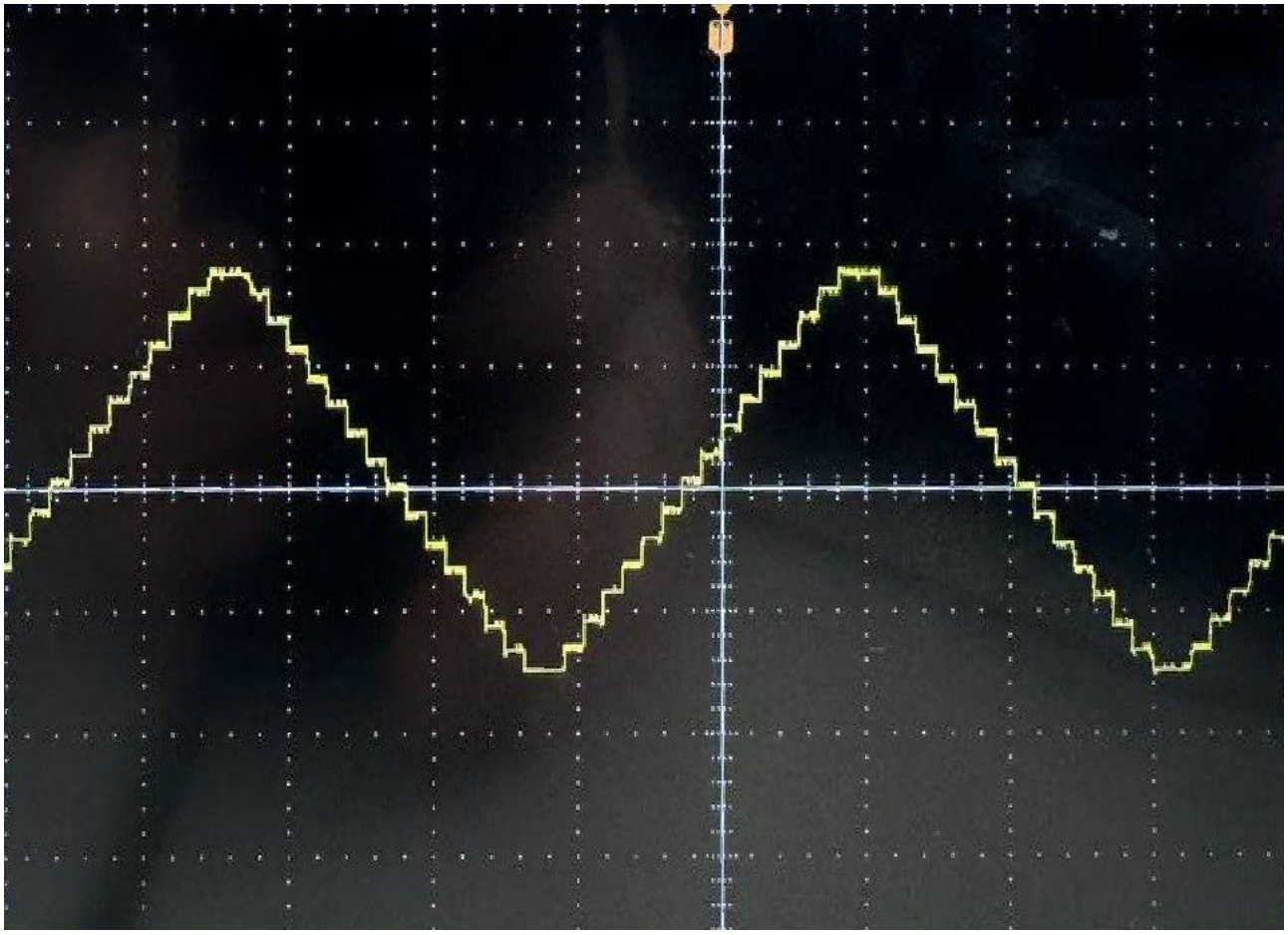
- Настраиваем периферийное устройство TIM
- Включить периферийный счетчик TIM

Работа с осциллографом:

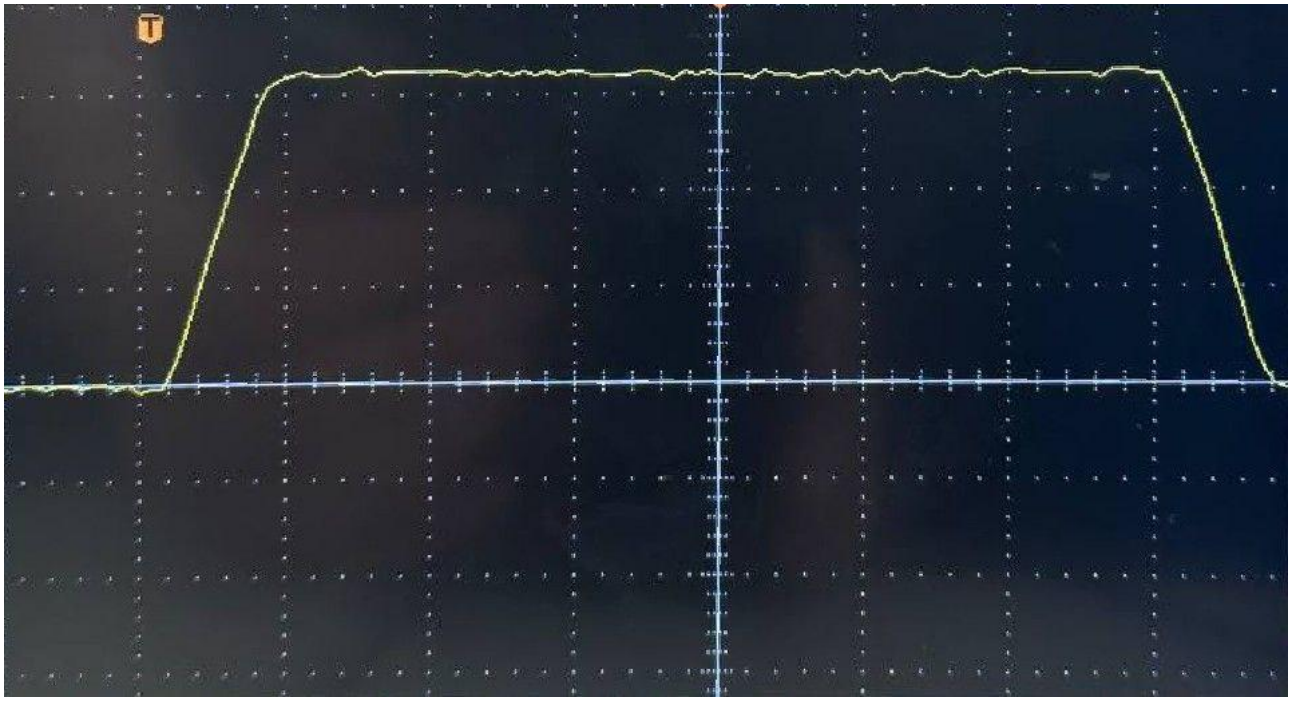
Прямолинейный сигнал с размахом 2.



Сигнал в форме синусоиды:



Изменение уровня сигнала:



Вывод:

Мы ознакомились с архитектурой низкоуровневых библиотек и промежуточного программного обеспечения микроконтроллера. Закрепили навыки работы с осциллографом и оценочной платой MCBSTM32F200 в качестве измерительного генератора.