САНКТ-ПЕТЕРБУРСКИЙ

УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Отчет по лабораторной работе

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

Работу выполнил:

Янковский Э. А.

Студент группы 5130201/30003

Преподаватель:

Вербова Н. М.

1. Тема и цель работы

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 в качестве измерительного генератора.

1. Задание на лабораторную работу

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).

* 1. Теоретическая сводка

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 c 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\_StdPeriph\_Lib\_VX.Y.Z, которая содержит следующие подпапки:

A screenshot of a computer program

AI-generated content may be incorrect.

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]](#footnote-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.

1. Алгоритм 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[32] = {0x0, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xaa, 0xbb, 0xCC, 0xdd, 0xee, 0xFF, 0xff, 0xee, 0xdd, 0xcc, 0xbb, 0xaa, 0x99, 0x88, 0x77, 0x66, 0x55, 0x44, 0x33, 0x22, 0x11, 0x00};

/\* 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, 32, 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);

}

* 1. Дополнительные комментарии к коду

Сначала подключаем файл, который содержит все прототипы функций для модуля драйверов HAL (Hardware Abstraction Layer – уровень абстракции аппаратного обеспечения, позволяющий управлять различными регистрами и характеристиками чипа STM32F2xx). Затем определяем макросы и глобальные переменные, включая массив для создания волнообразного сигнала (синусоиды).

Объявляем прототипы функций для работы с АЦП и таймерами.

В функции void HAL\_DAC\_MspInit (DAC\_HandleTypeDef\* hdac) активируем часы ЦАП, где происходит первичная деинициализация DAC MSP – освобождаются аппаратные ресурсы:

- Отключение часов периферийного устройства

- Возврат GPIO в исходное состояние

Настраиваем параметры для первого канала и связываем инициализированный дескриптор DMA с дескриптором DAC.

Следующая функция выполняет краткую инициализацию TIM MSP, настраивая аппаратные ресурсы:

- Включение часов периферийного устройства

- Конфигурация GPIO периферии

- Указатель на дескриптор TIM

В основном блоке программы выполняется:

- Настройка предварительной выборки флэш-памяти, инструкций и кешей данных

- Конфигурация Systick для генерации прерывания каждые 1 мс

- Установка приоритета группы NVIC на 4

- Глобальная инициализация MSP (пакет поддержки MCU)

Для генерации волнообразного сигнала в функции main вызывается функция генерации сигнала, после чего запускается бесконечный цикл while(1). На этом основная часть функции main завершается.

Затем создается функция SystemClock\_Config, в которой:

- Включается осциллятор HSE и активируется PLL с HSE в качестве источника, параметры устанавливаются согласно методическим указаниям

- Выбирается PLL как источник системного таймера и настраиваются HCLK, PCLK1 и PCLK2 – делители часов.

Далее определяем функцию DAC\_Ch1\_WaveConfig, в которой:

- Инициализируется периферийное устройство ЦАП

- Конфигурируется первый канал ЦАП

- Включается канал 1 DAC и связанный с ним DMA

- Включается DAC Channel1

- Устанавливается регистр DHR12RD для канала 1 ЦАП

В завершение переопределяем функцию TIM6\_Config, которая отвечает за конфигурацию TIM6:

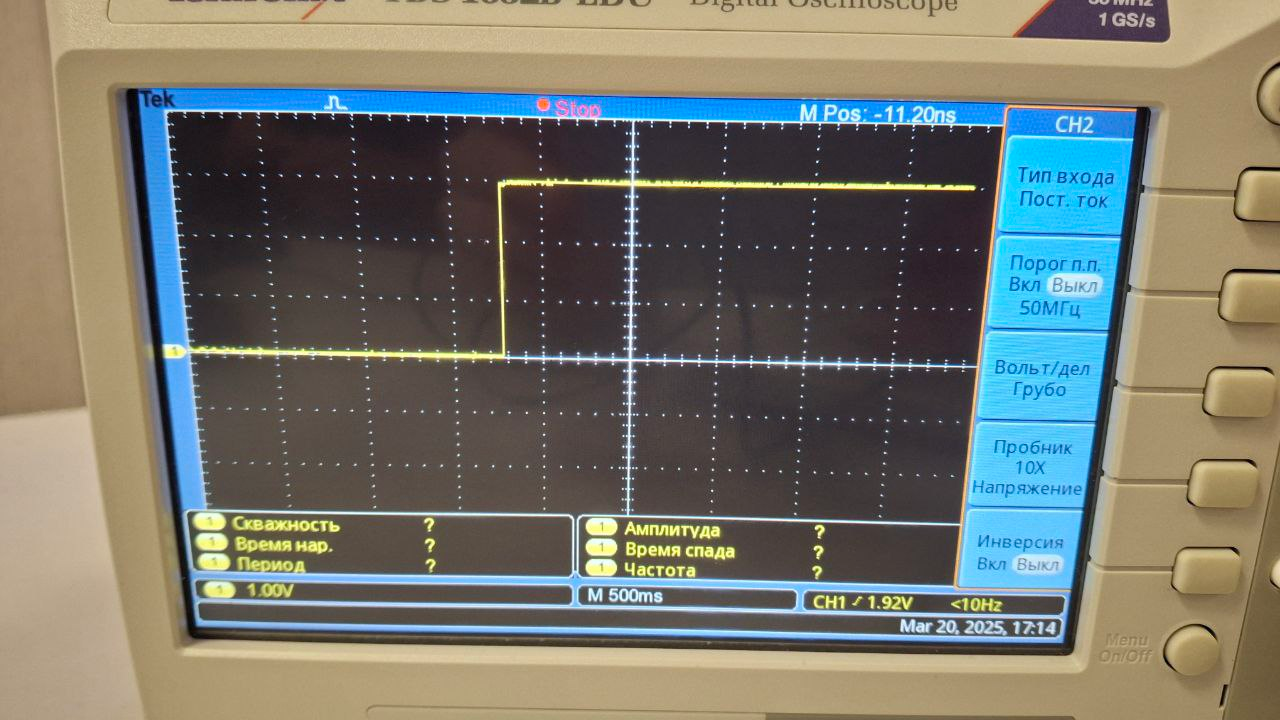
- Настраивается периферийное устройство TIM

- Включается периферийный счетчик TIM.

Для изменения сигнала на синусоидальный был использован массив (матрица) размером в 32 элементов, 16 последних из которых являются зеркальным отражением 16 первых.

1. Полученные результаты

Прямолинейный сигнал с размахом 3.



Периодический ступенчатый сигнал из 6 ступеней из методических материалов.

Размах напряжения: 3.2V

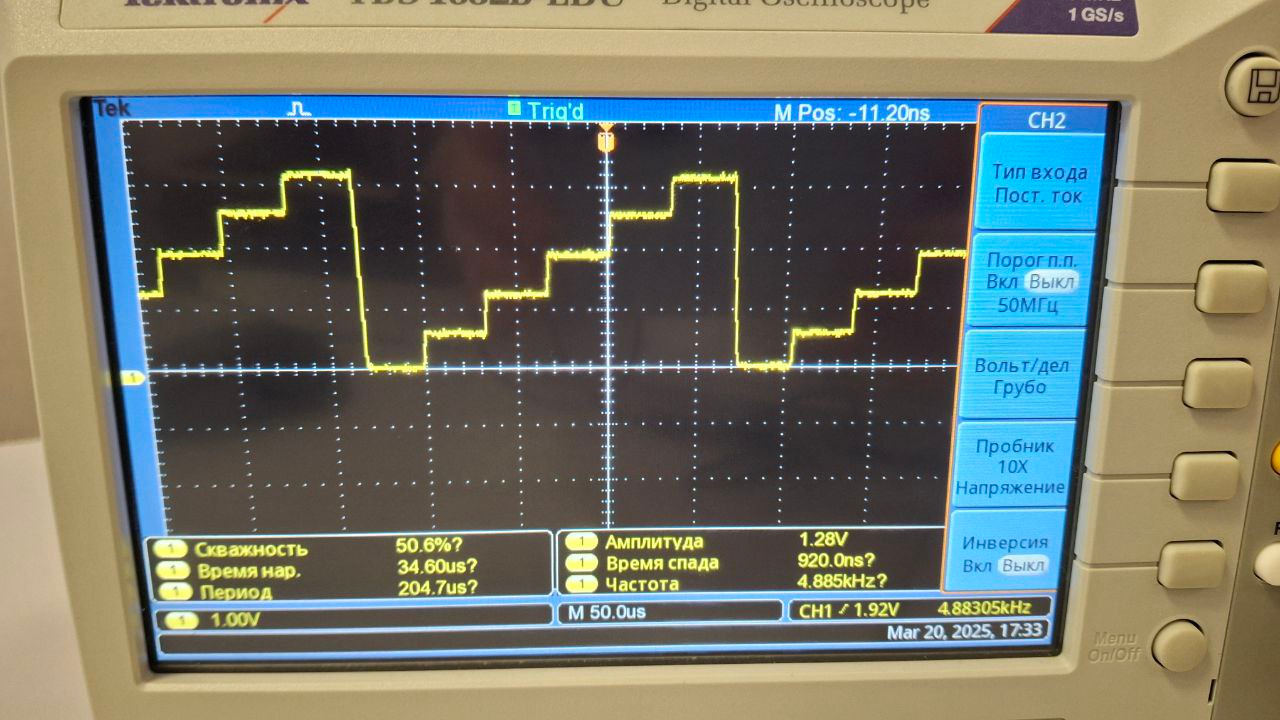
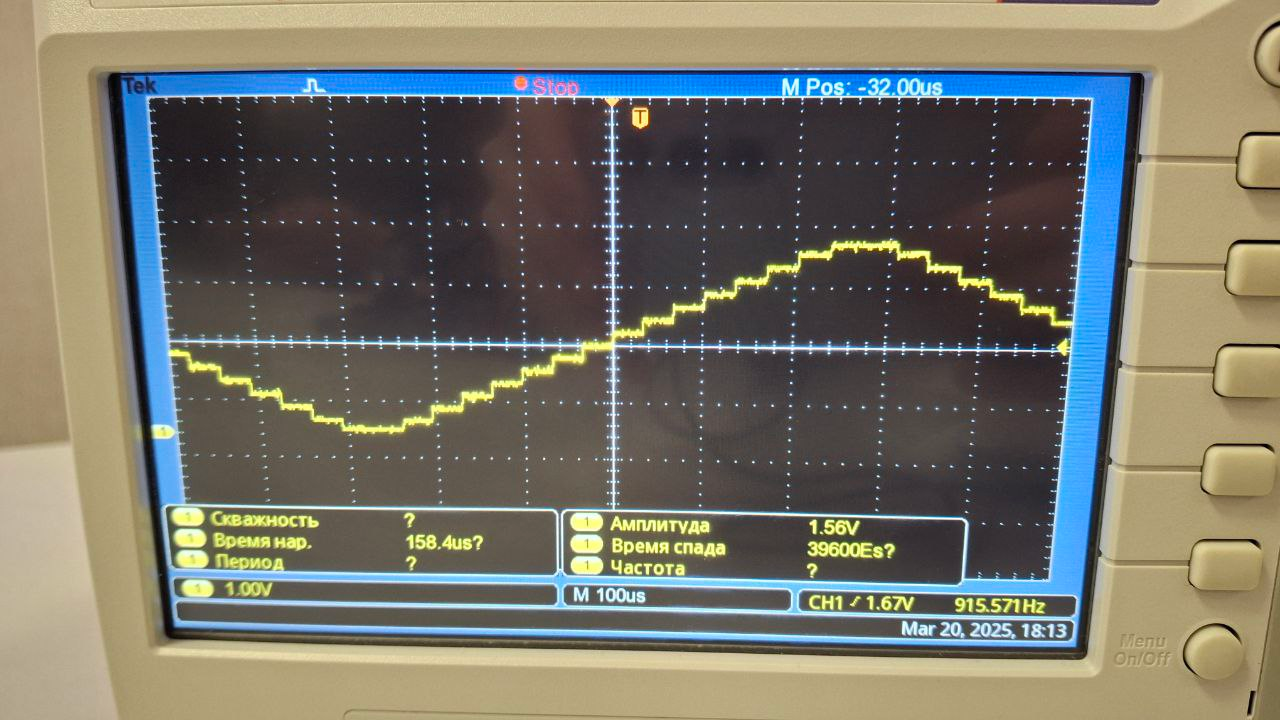
Период: 204.7us

График осциллографа после модификации программы – синусоидальный сигнал.

Амплитуда: 1.72V

Период: 1.092ms

Размах: 3.2V



1. Вывод

Мы изучили архитектуру низкоуровневых библиотек и промежуточного программного обеспечения микроконтроллера. Также практиковались в использовании осциллографа и оценочной платы MCBSTM32F200 в роли измерительного генератора.

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. [↑](#footnote-ref-1)