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 № 6.

Работу выполнила:

Гусева С.А.

студент группы 3530201/10001

Проверила:

Вербова Н. М.

Санкт-Петербург - 2023 г.

**Тема**:

Управление аппаратными таймерами STM32F200

**Цель:**

Закрепить навыки работы с низкоуровневыми библиотеками и промежуточным программным обеспечением микроконтроллера. Ознакомится со способами управления аппаратными таймерами STM32F200. Ознакомиться с приемами отладки программ.

**Постановка задачи**:

используя библиотеки Keil μVision5, разработать программу для микроконтроллера (МК) STM32F200, которая включает и выключает светодиоды: один при достижении содержимым таймера заданных значений, а другой при достижении заданных значений содержимым программного счетчика.

**Теоретические данные:**

Аппаратный таймер по существу является независимым счетчиком, который с заданной скоростью считает от нуля до своего максимального значения, генерируя различные события. Он работает в фоновом режиме независимо от написанной Вами на C/C++ программы, а его значения обычно соответствуют последовательности показанной ниже:

Chart, box and whisker chart

Description automatically generated

Подробное описание таймеров общего назначения (TIM2 – TIM5) используемых в микроконтроллере STM32F200 приведено в п. 14 справочного руководства (см. файл CD00225773.pdf).

Таймеры широко используются для отсчета временных интервалов, подсчета импульсов, могут применяться совместно с энкодерами (инкрементальные датчики поворота) и т.п. Одним из наиболее широких применений таймеров является их использование для получения сигналов с широтно-импульсной модуляцией (ШИМ).

Широтно-импульсная модуляция является основным видом модуляции, применяемым в устройствах управления силовой электроники, двигателями электромеханических и радиотехнических следящих систем, регуляторах возбуждения синхронных машин и т.п.

При ШИМ в соответствии с величиной управляющего напряжения модуляции изменяется ширина (длительность) импульсов последовательности.

Различают ШИМ I (первого рода), при которой длительность импульса пропорциональна значению модулирующего сигнала в момент среза импульса, и ШИМ II (второго рода), когда эта длительность пропорциональна значению модулирующего сигнала в тактовой точке (например, фронта импульса). Если за время, пока длится импульс, значение модулирующего сигнала меняется мало, различие между ШИМ I и ШИМ II незначительно. В тех случаях, когда положение фронта (или среза) импульсов фиксируется, а в соответствии с модулирующим напряжением меняется положение среза (или фронта) импульса, ШИМ называется односторонней.

ШИМ достигается путем использования управляемой временной задержки, где управляющим является напряжение модуляции (задающее (желаемое) напряжение). Ниже приведены временные диаграммы, иллюстрирующие процессы ШИМ для односторонней ШИМ первого рода при синусоидальной форме задающего напряжения:

Diagram, engineering drawing

Description automatically generated

**Код программы:**

#include "stm32f2xx\_hal.h"

GPIO\_InitTypeDef GPIO\_InitStruct;

TIM\_HandleTypeDef htim;

void InitializeLED()

{

RCC->AHB1ENR |= RCC\_AHB1ENR\_GPIOGEN;

/\* GPIO base configuration \*/

GPIO\_InitStruct.Pin = (GPIO\_PIN\_7);

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_LOW;

HAL\_GPIO\_Init(GPIOG, &GPIO\_InitStruct);

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_RESET);

}

void InitializeTimer()

{

RCC->APB1ENR |= RCC\_APB1ENR\_TIM2EN ;

/\* Time base configuration \*/

htim.Instance = TIM2;

htim.Init.Period = 0x500;

htim.Init.Prescaler = 40000;

htim.Init.ClockDivision = 0;

htim.Init.RepetitionCounter = 0;

htim.Init.CounterMode = TIM\_COUNTERMODE\_UP;

HAL\_TIM\_Base\_Init(&htim);

/\* Enable TIM peripheral counter \*/

HAL\_TIM\_Base\_Start(& htim);

}

int main()

{

InitializeLED();

InitializeTimer();

for (;;)

{

int timerValue = TIM2->CNT;

if (timerValue == 400)

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_SET);

if (timerValue == 420)

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_RESET);

if (timerValue == 440)

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_SET);

if (timerValue == 460)

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_RESET);

if (timerValue == 480)

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_SET);

else if (timerValue == 500)

HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_7,

GPIO\_PIN\_RESET);

}

}

**Алгоритм программы:**

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

Введем одну частную переменную для портов ввода-вывода общего назначения GPIO и одну частную переменную для таймера TIM. Первая переменная для определения структуры инициализации портов ввода-вывода общего назначения GPIO, а вторая для определения постоянной структуры конфигурации таймера.

В соответствии с п. 12.11.2 обзора работы с периферией системы (см. файл DM00105879.pdf) нужно составить функцию инициализации портов ввода вывода общего назначения. Функция инициализации портов ввода вывода общего назначения. В ней мы включаем тактирования порта G и настраиваем параметры работы GPIO.

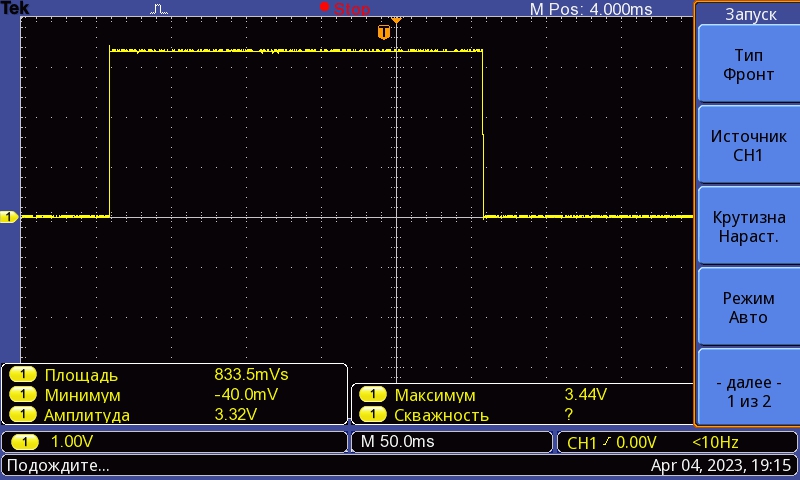
Теперь в соответствии с пунктами 58.2.13 и 58.2.17 описания драйвера таймера (см. файл DM00105879.pdf) составим функцию инициализации таймера:

Включение тактирования таймера и выставляем параметры аналогично предыдущей лабораторной работе.

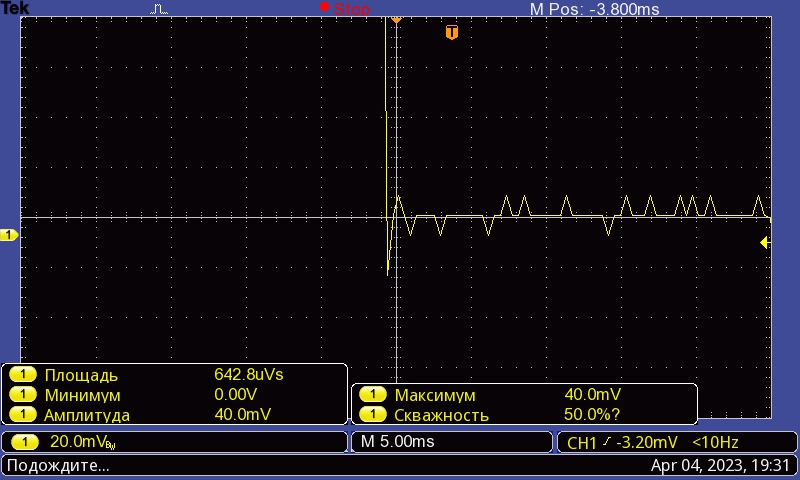
И наконец, последняя функция – основное тело программы. Здесь нужно инициализировать порты ввода-вывода и таймер. После чего, в бесконечном цикле выполнять проверку содержимого таймера (“чтение на лету”) и при равенстве содержимого заданным значениям включать или выключать светодиод. Все это будет выглядеть следующим образом:

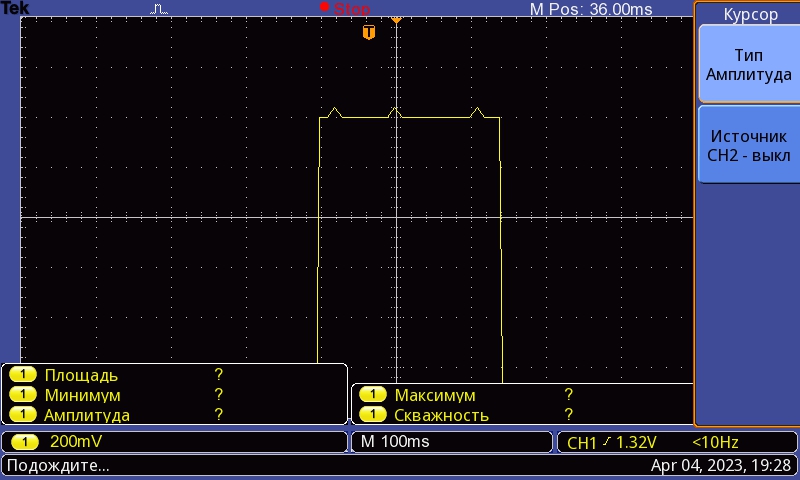
**Работа с осциллографом:**

Площадь ограниченная сигналом = 5\*50ms \* 3V = 750ms\*V, минимум и максимум:

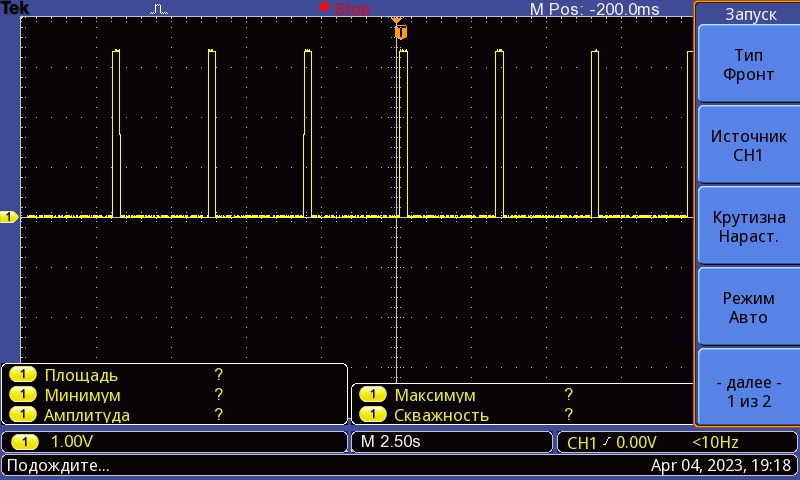


Перерегулирование сигнала:

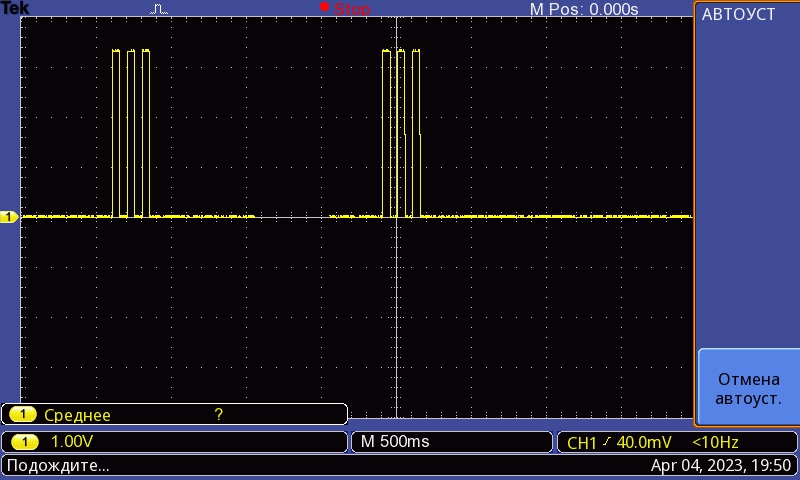




Количество импульсов = 7, нарастающих фронтов фронтов = 7, падающих фронтов = 6.



Ширина пакета сигнала (пачки импульсов) = 0.6\*500ms = 1.2ms:



**Вывод:**

Были закреплены навыки работы с низкоуровневыми библиотеками и промежуточным программным обеспечением микроконтроллера. Проведено ознакомление со способами управления аппаратными таймерами STM32F200 и с приемами отладки программ.