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 № 7.

Работу выполнила:

Гусева С.А.

студент группы 3530201/10001

Проверила:

Вербова Н. М.

Санкт-Петербург - 2023 г.

**Тема**:

Использование таймеров STM32F200 в режиме сравнивания выходных величин.

**Цель:**

Ознакомиться с конфигурированием аппаратных таймеров STM32F200. Закрепить навыки работы с низкоуровневыми библиотеками и промежуточным программным обеспечением микроконтроллера. Закрепить навыки отладки программ.

**Постановка задачи**:

используя библиотеки Keil μVision5, разработать программу для микроконтроллера (МК) STM32F200, которая выдает выходные импульсы на разных выводах порта при достижении содержимым таймера заданных значений.

**Теоретические данные:**

Аппаратные таймеры микроконтроллера STM32F200 представляют собой отдельные аппаратные блоки, которые могут выполнять счет от 0 до заданного значения, в промежутке вызывая некоторые события. В режиме сравнивания содержимое регистра предварительной загрузки копируется в теневой регистр, содержимое которого сравнивается с содержимым счетчика. В этом режиме таймер может управлять одним или несколькими выходными каналами. Когда содержимое счетчика достигает значения нуля, максимума или заданного для каждого из каналов своего значения сравниваемой величины значение выходного канала может быть изменено. Какие события изменяют значение, и как оно изменяется, определяют разнообразные варианты конфигурации.

Shape

Description automatically generated

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

**Код программы:**

#include "stm32f2xx\_hal.h"

GPIO\_InitTypeDef GPIO\_InitStruct;

TIM\_HandleTypeDef htim;

TIM\_OC\_InitTypeDef outputChannelInit;

void InitializeGPIO()

{

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

/\* GPIO base configuration \*/

//GPIO\_InitStruct.Pin = ();

GPIO\_InitStruct.Pin = (GPIO\_PIN\_12|GPIO\_PIN\_13);

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

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

GPIO\_InitStruct.Alternate = GPIO\_AF2\_TIM4;

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

}

void InitializeTimer()

{

RCC->APB1ENR |= RCC\_APB1ENR\_TIM4EN ;

/\* Time base configuration \*/

htim.Instance = TIM4;

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);

}

void InitializePWMChannel2()

{

/\* Channel 2 base configuration \*/

outputChannelInit.OCMode = TIM\_OCMODE\_PWM1;

outputChannelInit.Pulse = 5

00;

outputChannelInit.OCFastMode = TIM\_OCFAST\_DISABLE;

outputChannelInit.OCPolarity = TIM\_OCPOLARITY\_HIGH;

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

HAL\_TIM\_OC\_ConfigChannel(&htim, &outputChannelInit, TIM\_CHANNEL\_2);

/\* Enable TIM peripheral counter \*/

HAL\_TIM\_OC\_Start(&htim, TIM\_CHANNEL\_2);

}

void InitializePWMChannel()

{

/\* Channel base configuration \*/

outputChannelInit.OCMode = TIM\_OCMODE\_PWM1;

outputChannelInit.Pulse = 400;

outputChannelInit.OCFastMode = TIM\_OCFAST\_DISABLE;

outputChannelInit.OCPolarity = TIM\_OCPOLARITY\_HIGH;

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

HAL\_TIM\_OC\_ConfigChannel(&htim, &outputChannelInit, TIM\_CHANNEL\_1);

/\* Enable TIM peripheral counter \*/

HAL\_TIM\_OC\_Start(&htim, TIM\_CHANNEL\_1);

}

int main()

{

InitializeGPIO();

InitializeTimer();

InitializePWMChannel();

InitializePWMChannel2();

for (;;)

{

}

}

**Алгоритм 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.

Пишем функцию InitializeGPIO, в которой мы:

-Указываем, что будем настраивать пины PD12 и PD13

-режим работы — выход двумя состояниями (*Push-Pull*)

-настраиваем скорость работы выхода

-настраиваем работу с базовым таймером TIM4

Пишем функцию InitializeTimer, в которой мы устанавливаем параметры таймера: период, предварительный делитель, тактовое деление, счетчик повторений, режим счетчика.

Далее пишем две функции, в которых подключаем первую линию к таймеру и определяем параметры канала: режим, частоту и тд.

В теле функции main вызываем четыре функции, перечисленные выше, и запускаем бесконечный цикл for.

**Работа с осциллографом:**

В работе с осциллографом мы меняли режимы синхронизации двух сигналов с различной шириной.

На изображении представлена синхронизация по фронту, есть возможность выбора синхронизации по нарастающему и нисходящему фронтам.

A screen shot of a computer

Description automatically generated with low confidence

На изображении представлена синхронизация по импульсу, есть возможность выбора синхронизаци по таким логическим операциям, как: «=», «!=», «>», «<».

A screenshot of a computer

Description automatically generated with medium confidence

**Вывод:**

Мы ознакомились с конфигурированием аппаратных таймеров STM32F200. Закрепили навыки работы с низкоуровневыми библиотеками и промежуточным программным обеспечением микроконтроллера, а также закрепили навыки отладки программ.