МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационная безопасность систем и технологий»

Отчет

по лабораторной работе №2

на тему «Таймеры и обработка прерываний микроконтроллеров STM32»

Дисциплина: ПМК

Группа: 21ПИ1

Выполнил: Гусев Д. А.

Количество баллов:

Дата сдачи:

Принял: Хворостухин С. П .

2024

1 Цель работы: Ознакомиться с программными средствами работы с таймерами и прерываниями микроконтроллеров STM32.

2 Задание на лабораторную работу.

2.1 Получить вариант задания у преподавателя.

2.2 Рассчитать значение предделителя каждого таймера.

2.3 Создать проект в среде Keil uVision5 для микроконтроллера STM32F103RB.

2.4 Выбрать программные компоненты: CMSIS/Core; Device/Startup; Device/StdPeriph Drivers/Framework; Device/StdPeriph Drivers/GPIO; Device/StdPeriph Drivers/RCC; Device/StdPeriph Drivers/TIM.

2.5 Выполнить настройку режима отладки для проекта.

2.6 Разработать файл сценария.

2.7 Разработать программу согласно варианту задания, реализующую: инициализацию портов ввода-вывода; инициализацию таймера в режиме захвата сигнала; инициализацию таймера в режиме широтно-импульсной модуляции; обработку прерываний таймеров.

2.8 Выполнить симуляцию разработанной программы с использованием функций отладки. Зафиксировать параметры обрабатываемых и формируемых сигналов.

2.9 Рассчитать коэффициент заполнения по полученным сигналам широтно-импульсной модуляции.

2.10 Сделать выводы по проделанной работе и оформить отчет.

3 Выполнение лабораторную работы:

3.1 Был получен вариант задания – 8. Данные для варианта заданя представлены в таблице 1.

Таблица 1 — Вариант задания №8

Номер варианта	Таймер захвата сигнала			Таймер ШИМ		
	Номер таймера	Номер канала	Значение отсчета	Номер таймера	Номер канала	Значение отсчета

8	TIM3	CH1	5 мкс	TIM2	CH4	100 мкс

3.2 Было рассчитано значение предделителя каждого таймера с учетом требования, что значение предделителя, записываемое в регистр таймера, должно быть на 1 меньше рассчитанного.

Prescaler = $F_{TIM} \times T_{CNT}$ - 1, где F_{TIM} - частота тактового сигнала, Гц; T_{CNT} - время одного отсчета таймера, с.

Согласно варианту F_{TIM} = 108MHz = 108 * 10^6 Гц, T_{CNT} = 5 мкс = 5*10^(-6) сек → *Capture Prescaler* = 539.

F_{TIM} = 108MHz = 108 * 10^6 Гц, T_{CNT} = 100 мкс = 100*10^(-6) сек → *PWM Prescaler* = 10799.

3.3 Был создан проект в среде Keil uVision5 для микроконтроллера STM32F103RB. Результат представлен на рисунке 1.

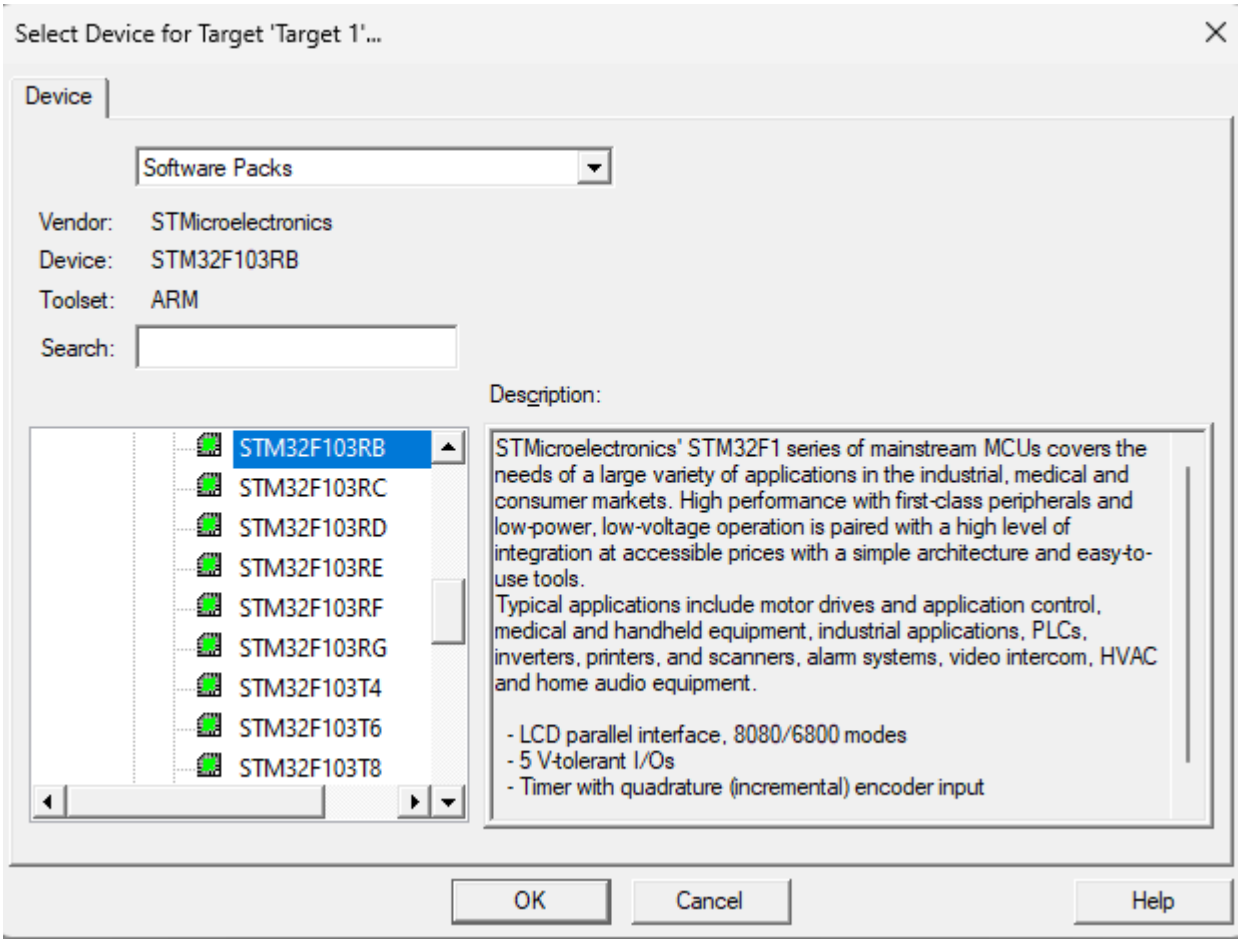

Рисунок 1 — Создание проекта для микроконтроллера STM32F103RB

3.4 Были выбраны программные компоненты: CMSIS/Core; Device/Startup; Device/StdPeriph Drivers/Framework; Device/StdPeriph Drivers/GPIO; Device/StdPeriph Drivers/RCC; Device/StdPeriph Drivers/TIM. Результат представлен на рисунке 2.

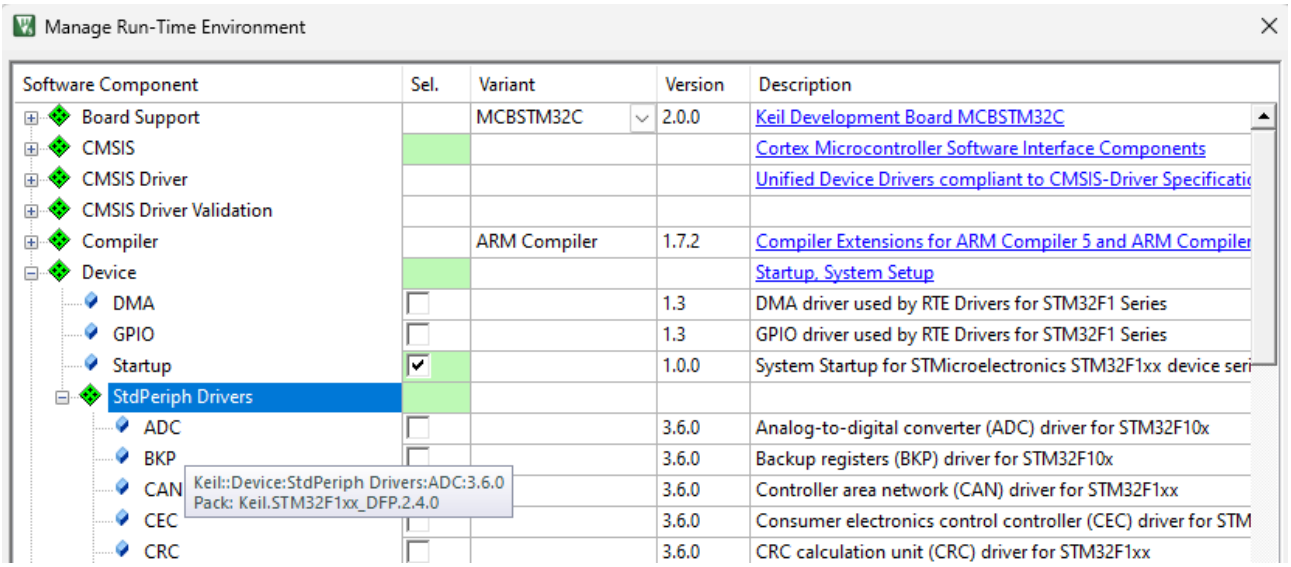

Рисунок 2 — Выбор компонентов

3.5 Была выполнена настройка режима отладки для проекта. Для этого в папке проекта был создан файл MAP.ini со следующим содержанием:

```
MAP 0x40000000, 0x47FFFFFF READ WRITE;
```

Далее была открыта вкладка «Option for Target...» затем была открыта вкладка «Debug», был включен переключатель «UseSimulator». В поле «DialogDLL» было записано DARMSTM.DLL, в поле «Parameter» было записано: - pSTM32F103RB. Был установлен путь к файлу MAP.ini в поле «InitializationFile». Результат представлен на рисунке 3.

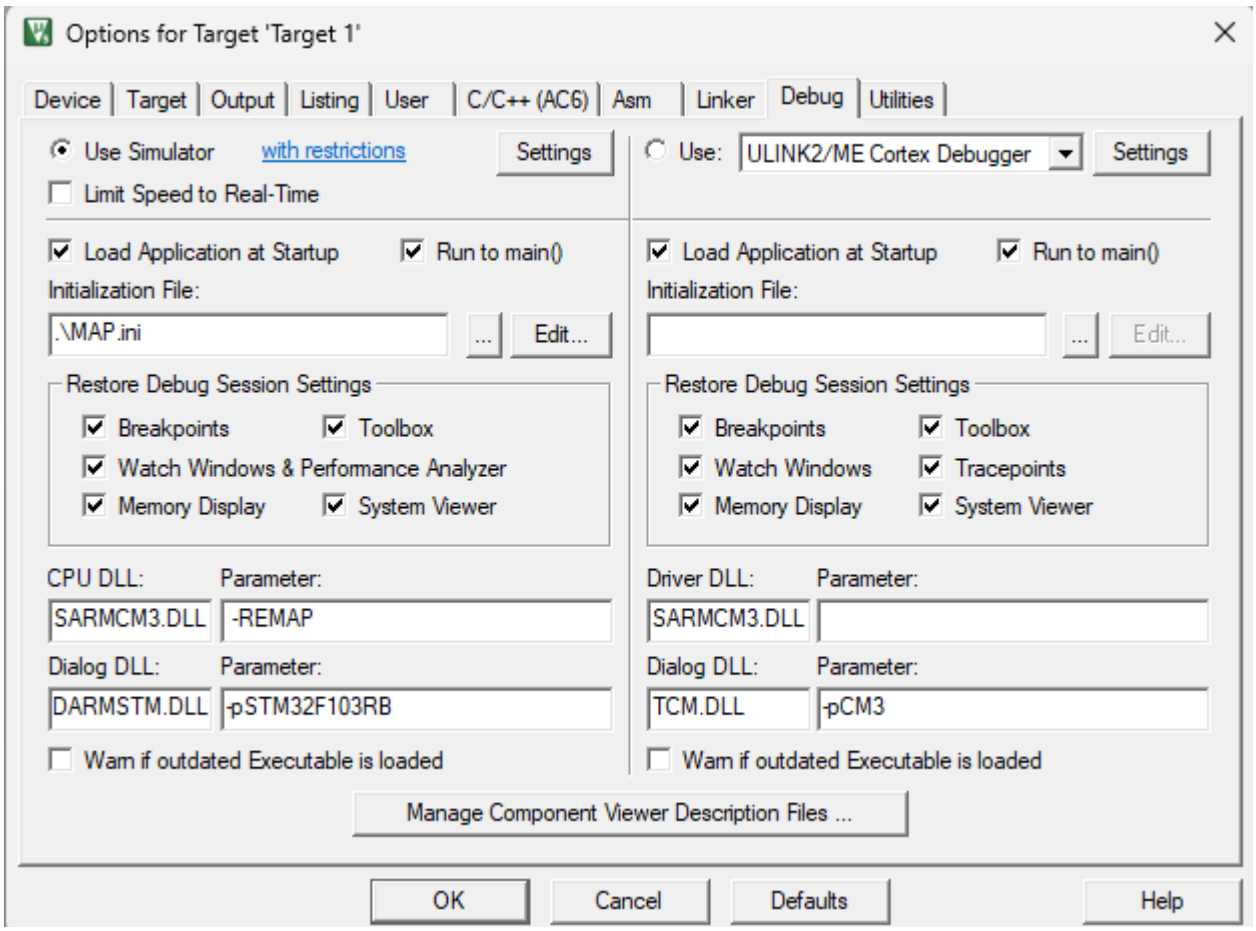

Рисунок 3 — Настройка проекта

3.6 Был разработан файл сценария. Код файла представлен ниже:

```
signal void input() {
  int i, j, delay, sleeps;
  // period array 2, 4, 6, 12, 20
  sleeps = 0xA6321;
  while (1) {
    for (j = 0; j < 6; j++) {
      delay = (sleeps >> (j * 4)) & 0xF;
      for (i = 0; i < 20; i++) {
        PORTA |= 0x40;
        swatch(delay * 0.001);
        PORTA &= ~0x40;
        swatch(delay * 0.001);
      }
    }
  }
}
```

3.7 Была разработана программа, реализующая: инициализацию портов ввода-вывода; инициализацию таймера в режиме захвата сигнала; инициализацию таймера в режиме широтно-импульсной модуляции; обработку прерываний таймеров. Код программы представлен в Приложении А.

3.8 Была выполнена симуляция разработанной программы с использованием функций отладки. Результаты представлены на рисунках 4 — 5.

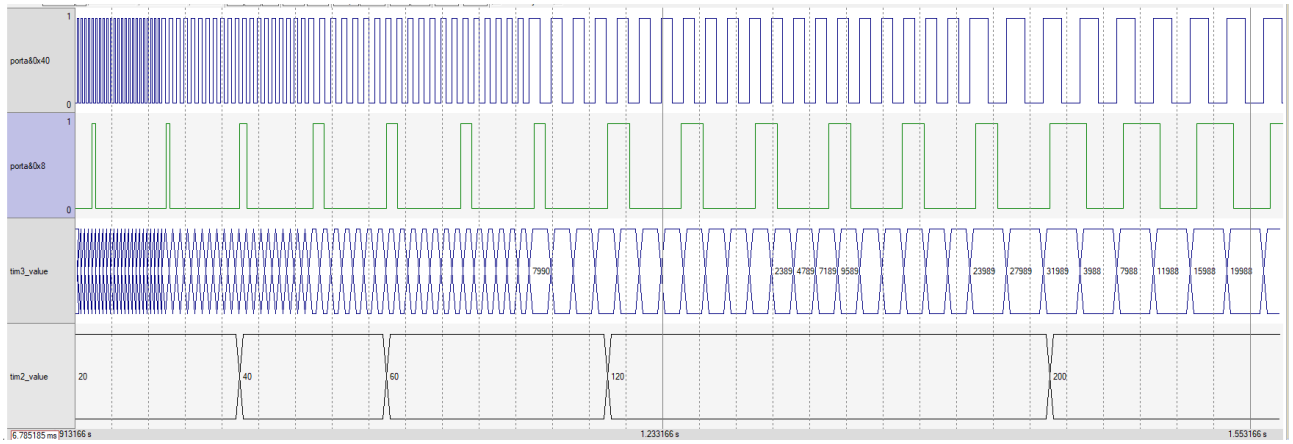

Рисунок 4 - Симуляция

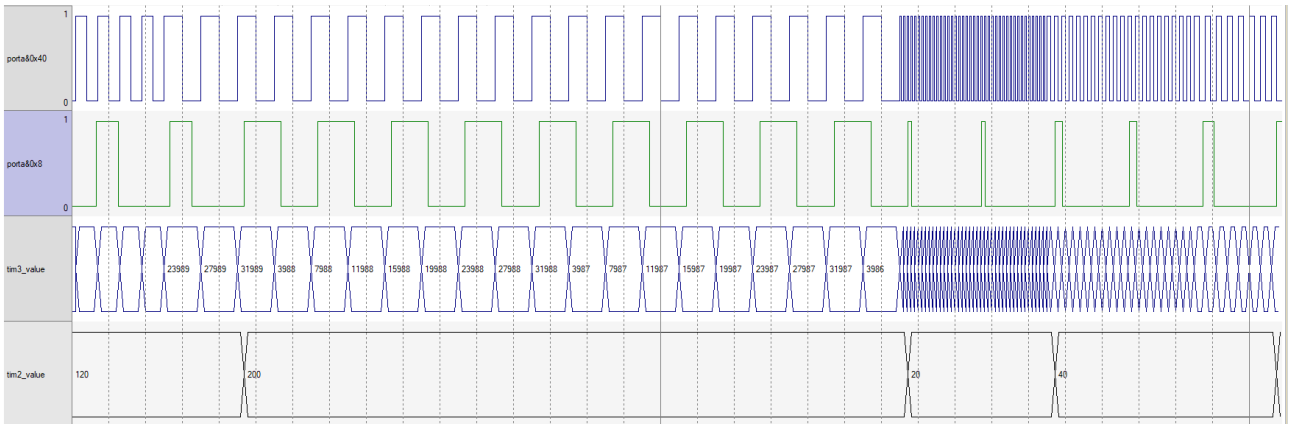

Рисунок 5 - Симуляция

3.9 Был рассчитан коэффициент заполнения по полученным сигналам широтно-импульсной модуляции. Результат представлен в таблице 2.

Таблица 2 — Коэфециэты заполнения.

Период выходного сигнала, мс	40
Ширина выходного сигнала, мс	Коэф D
2	5,00%
4	10,00%

6	15,00%
12	30,00%
20	50,00%

4 Вывод: было выполнено ознакомление с программными средствами работы с таймерами и прерываниями микроконтроллеров STM32.

Приложение А

Код программы

```
#include "stm32f10x.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_tim.h"

/* Константы для настройки таймеров */
static uint16_t CAPTURE_PRESCALER = 540;
static uint16_t CAPTURE_PERIOD = 32000;
static uint16_t PWM_PRESCALER = 10800;
static uint16_t PWM_PERIOD = 400; /* 200 (max input period) * 2  */
static uint16_t SYNC = 20;        /* PWM_PRESCALER / CAPTURE_PRESCALER */

/* Хранение значения таймера */
uint16_t tim3_value = 0;
uint16_t tim2_value = 0;
uint16_t pwm_pulse = 0; /* 2ms default */

/* Функция инициализации портов */
static void initGPIO(void)
{
  /* Объявление структуры для инициализации порта */
  GPIO_InitTypeDef PORT;
  /* Включение тактирования порта A */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

  /* Настройка порта PA6 */
  PORT.GPIO_Pin = GPIO_Pin_6;             /* Настройка пина 6 */
  PORT.GPIO_Mode = GPIO_Mode_IN_FLOATING; /* Режим входа, подтяжка отключена */
  PORT.GPIO_Speed = GPIO_Speed_2MHz;      /* Скорость порта 2 МГц */
  GPIO_Init(GPIOA, &PORT);                /* Применение настроек к порту A */

  /* Настройка порта PA3 */
  PORT.GPIO_Pin = GPIO_Pin_3;        /* Настройка пина 3 */
```

```
  PORT.GPIO_Mode = GPIO_Mode_AF_PP;  /* Режим альтернативной функции, push-pull */
  PORT.GPIO_Speed = GPIO_Speed_2MHz; /* Скорость порта 2 МГц */
  GPIO_Init(GPIOA, &PORT);           /* Применение настроек к порту A */
}

static void initTIM3(void)
{
  /* Объявление структуры для инициализации канала таймера */
  TIM_ICInitTypeDef CHANEL;
  /* Объявление структуры для инициализации таймера */
  TIM_TimeBaseInitTypeDef TIMER;

  /* Включение тактирования таймера TIM3 */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  /* Настройка периода и прескеллера TIM3 */
  TIMER.TIM_Period = CAPTURE_PERIOD;
  TIMER.TIM_Prescaler = CAPTURE_PRESCALER - 1;

  /* Настройка делителя частоты, режим счета таймера TIM3 */
  TIMER.TIM_ClockDivision = 0;                /* Делитель частоты таймера */
  TIMER.TIM_CounterMode = TIM_CounterMode_Up; /* Режим счета вверх */
  TIM_TimeBaseInit(TIM3, &TIMER);             /* Применение настроек */

  /* Настройка первого канала */
  CHANEL.TIM_Channel = TIM_Channel_1;
  CHANEL.TIM_ICPolarity = TIM_ICPolarity_Rising;     /* Передний фронт */
  CHANEL.TIM_ICSelection = TIM_ICSelection_DirectTI; /* Выбор входа - прямой вход */
  CHANEL.TIM_ICPrescaler = TIM_ICPSC_DIV1;           /* Предделитель входного сигнала
*/
  CHANEL.TIM_ICFilter = 0x0;                         /* Фильтр входного сигнала */
  TIM_ICInit(TIM3, &CHANEL);                         /* Применение настроек к каналу 1
таймера TIM3 */
  TIM_ITConfig(TIM3, TIM_IT_CC1, ENABLE);            /* Включение прерывания по событию
захвата на канале 1 */

  TIM_Cmd(TIM3, ENABLE);     /* Включение таймера TIM3 */
  NVIC_EnableIRQ(TIM3_IRQn); /* Разрешить прерывания от таймера 3 */
```

```
}

/* Обработчик прерывания таймера TIM3 */
void TIM3_IRQHandler(void)
{
  uint16_t current_value;
  uint16_t current_pulse;
  /* Проверка наличия прерывания по событию захвата на канале 1 */
  if (TIM_GetITStatus(TIM3, TIM_IT_CC1) == RESET)
  {
    return; /* Если прерывания нет, выходим из обработчика */
  }

  /* Сброс флага прерывания */
  TIM_ClearITPendingBit(TIM3, TIM_IT_CC1);

  current_value = TIM_GetCapture1(TIM3);

  /* Если текущее значение захвата меньше сохраненного, обновляем сохраненное значение
и выходим */
  if (current_value < tim3_value)
  {
    tim3_value = current_value;
    return;
  }

  /* Определение периода */
  current_pulse = current_value - tim3_value;
  if (current_pulse != pwm_pulse)
  {
    pwm_pulse = current_pulse;
    /* Установка нового значения сравнения */
    TIM_SetCompare4(TIM2, pwm_pulse / SYNC);
  }

  /* Обновление сохраненного значения захвата */
  tim3_value = current_value;
}
```

```
/* Функция инициализации таймера TIM2 в режиме PWM */
static void initTIM2(void)
{
  /* Объявление структуры для инициализации канала таймера */
  TIM_OCInitTypeDef CHANNEL;
  /* Объявление структуры для инициализации таймера */
  TIM_TimeBaseInitTypeDef TIMER;

  /* Включение тактирования таймера TIM2 */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  /* Настройка периода и прескеллера TIM2 */
  TIMER.TIM_Period = PWM_PERIOD;
  TIMER.TIM_Prescaler = PWM_PRESCALER - 1;

  /* Настройка делителя частоты, режим счета таймера TIM2 */
  TIMER.TIM_ClockDivision = 0;                /* Делитель частоты таймера */
  TIMER.TIM_CounterMode = TIM_CounterMode_Up; /* Режим счета вверх */
  TIM_TimeBaseInit(TIM2, &TIMER);             /* Применение настроек */

  /* Настройка канала 4 в режиме PWM */
  CHANNEL.TIM_OCMode = TIM_OCMode_PWM1;             /* Режим работы канала - PWM1 */
  CHANNEL.TIM_OutputState = TIM_OutputState_Enable; /* Включение выхода канала */
  CHANNEL.TIM_OCPolarity = TIM_OCPolarity_High;     /* Полярность выходного сигнала -
прямая */
  TIM_OC4Init(TIM2, &CHANNEL);                      /* Применение настроек к каналу 4
таймера TIM2 */
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);        /* Включение прерывания на канале 4
*/
  TIM_SetCompare4(TIM2, pwm_pulse / SYNC);          /* установки нового значения
сравнения */

  TIM_Cmd(TIM2, ENABLE);     /* Включение таймера TIM2 */
  NVIC_EnableIRQ(TIM2_IRQn); /* Разрешить прерывания от таймера 2 */
}

/* Обработчик прерывания таймера TIM2 */
void TIM2_IRQHandler(void)
```

```
{
  /* Если произошло прерывание по событию захвата на канале 4 */
  if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
  {
    TIM_ClearITPendingBit(TIM2, TIM_IT_Update); /* Сброс флага прерывания */
    tim2_value = TIM_GetCapture4(TIM2);         /* Запись значения таймера в переменную
*/
  }
}

int main(void)
{
  __enable_irq(); /* Разрешить прерывания */
  initGPIO();     /* Вызов функции инициализации порта */
  initTIM3();     /* Вызов функции инициализации таймера */
  initTIM2();     /* Вызов функции инициализации таймера */
  while (1)
  {
  }
}
```