Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Аппаратные платформы встраиваемых систем

Отчет по лабораторной работе №2
На тему **«Система тактирования, системные таймеры STM32»**

Работу выполнили:
Студенты гр. 13541/1 Онищенко Д.И.
Шаменов А.А.
Преподаватель: Васильев А.Е.

Санкт – Петербург
2019

Содержание

Цель работы:

Изучить систему тактирования микроконтроллеров STM32, а также особенности работы с таймерами для решения различных задач, основанных на соблюдении заданных частот по переключению линий выводов.

Подготовка к работе:

1. Подготовить проект в IARWE, согласно документу IAR Project for IAR SK Board
2. Ознакомиться со схемой платы IAR SK (STM32F407ZG-board-schematic.pdf)
3. Ознакомиться с документацией МК STM32F407 (STM32F4xx_RM.pdf)

Теоретическая информация:

Система тактирования

Практически все блоки микроконтроллера тактируются от линии системной тактовой частоты (SYSCLK), исключение составляют лишь блоки USB, RTC и IWDG. Все это, как и источники тактирования можно увидеть на структурной схеме системы формирования тактовых частот на рисунке 1.

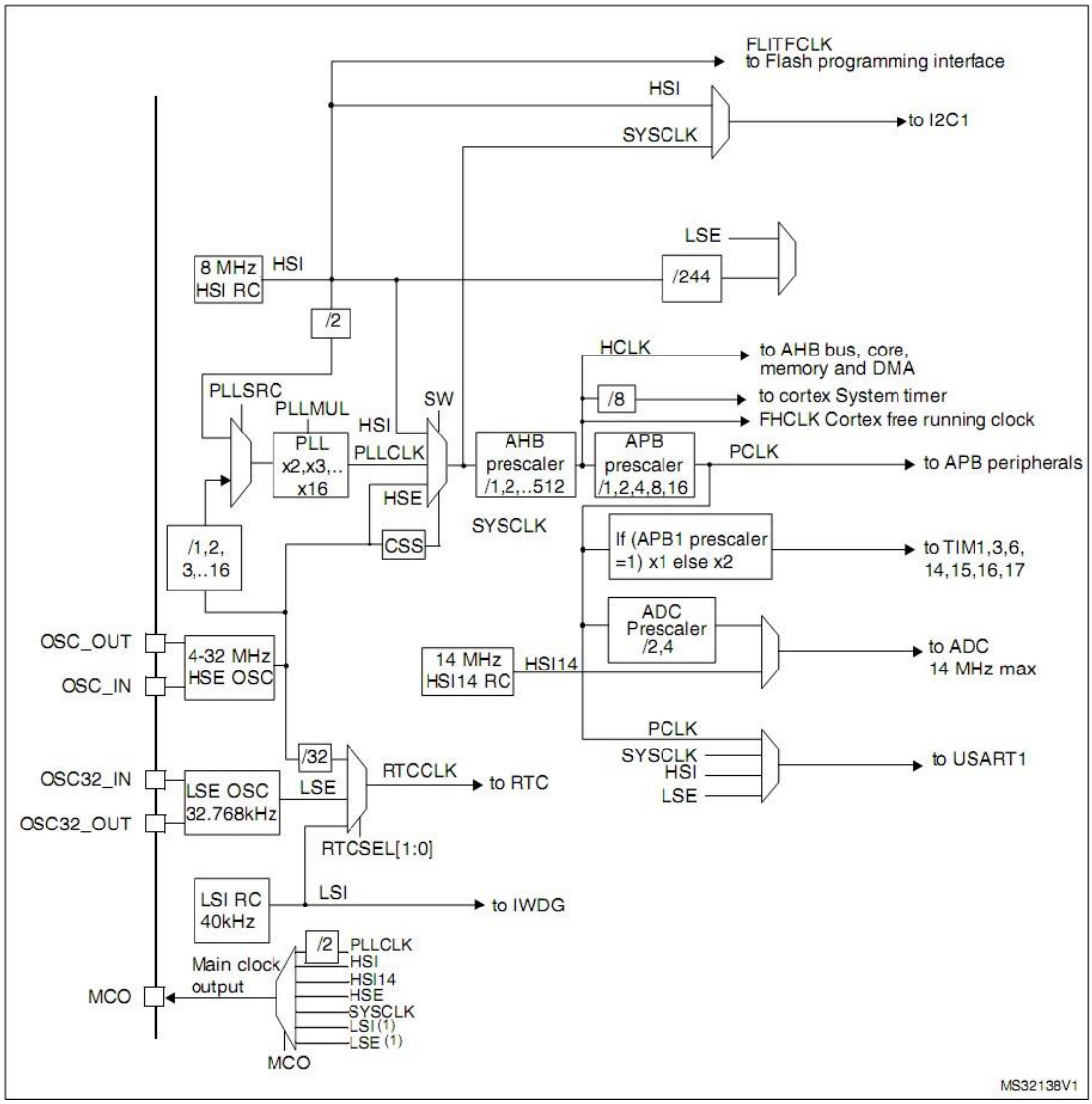


Рисунок 1 – Структурная схема системы формирования тактовых частот.

Одна из самых важных особенностей системы тактирования контроллера STM32, которую можно увидеть на приведенной схеме – источниками для системной тактовой частоты могут служить:

1. Генератор HSI – внутренний высокоскоростной (High Speed Internal)

2. Генератор HSE – внешний высокоскоростной (High Speed External)

3. Внутренний PLL – система фазовой автоподстройки частоты (Phased locked loop). Можно рассматривать этот вариант, как умножитель частоты с управляемым коэффициентом умножения.

Рассмотрим их более детально:

HSI

Встроенный в МК STM32F4 генератор HSI вырабатывает тактовую частоту 16 МГц, но для достижения более высокой частоты работы, частота сигнал может быть умножена на PLL. Генератор автоматически запускается при появлении питания Vcc. Первоначально процессорное ядро запускается на тактовой частоте HSI.

Преимущества:

- Быстрое время начала генерации тактовой частоты после подачи питания
- Отсутствие необходимости в использовании дополнительных электронных компонентов для работы микроконтроллера.

Недостатки:

- Низкая стабильность частоты генерируемого сигнала
- Мультипликативное накопление погрешности, при умножении на PLL

При работе с МК STM32, HSI является стандартным генератором тактовой частоты после включения, однако, следует учитывать, что при использовании библиотеки CMSIS, подключаемой для работы с периферией, источник SYSCLK переопределяется в файле system_stm32f4xx.c, код которого вызывается ASM-вставкой вашей программы, перед передачей управления в тело функции main.

Используя библиотеку SPL, сбросить все настройки системной тактовой частоты можно применяя следующую функцию (Все функции работы с тактированием находятся в файле stm32f4xx_rcc.h)

```
RCC_DeInit();
```

HSE

В качестве внешнего генератора могут выступать:

- Внешний тактовый сигнал не превышающий 25МГц, поданный на одну из ножек МК (OSC_IN), при этом ножка OSC_OUT должна находится в высокоимпендансном состоянии.
- Внешний кварцевый резонатор подключенный на ножки OSC_IN и OSC_OUT.

При использовании внешнего кварцевого резонатора достигается очень высокая стабильность частоты работы генератора.

Достоинствами использования HSE является то, что подобрать внешний генератор можно в соответствии с решаемыми задачами. Генерируемая частота и погрешность тактовой частоты, в этом случае, зависит от используемого генератора.

На плате IAR SK, используемой для лабораторного практикума, генератором HSE является внешний кварцевый резонатор, с частотой 25МГЦ. При использовании библиотеки CMSIS, именно он используется в качестве SYSCLK.

PLL

Внутренний умножитель частоты может служить источником системной тактовой частоты, умножая вошедший тактовый сигнал с одного из трех источников, на выбор:

- HSI/2
- HSE
- HSE/2

на множитель от 2-х до 16-ти. При умножении так же умножается ошибка тактовой частоты.

От системной тактовой частоты тактируются все элементы микроконтроллера, в т.ч. и таймеры, о которых пойдет речь в данной работе.

Таймеры

Всего в МК STM32 F4 14 таймеров и делятся они на 3 категории:

- Базовые таймеры (Basic Timers)
- Таймеры общего назначения (General Purpose Timers)
- «Продвинутые» таймеры (Advanced-control timers)

Основная задача данных таймеров – генерировать импульс прерывания по переполнению определенного значения, как показано на рисунке 2:

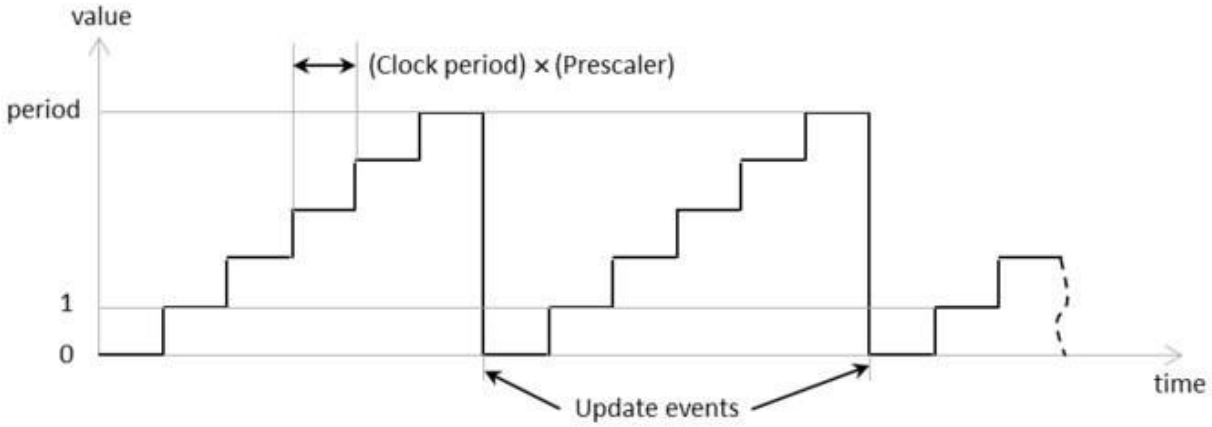


Рисунок 2 – Иллюстрация работы таймера

Характеристики таймеров представлены в таблице 1:

Таблица 1 – Характеристики различных таймеров:

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)
Advanced-control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	84	168
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	42	84
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	84	168
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	84	168
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	42	84
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	42	84
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	42	84

Ход работы:

Задание состоит из следующих пунктов:

1. Переключение светодиода с частотой 1КГц, система тактируется от внутреннего источника.

2. Переключение светодиода с частотой 1КГц, тактирование от внешнего источника.

3. Переключение светодиода с частотой 1КГц, тактирование от PLL.

4. Составить программу генерации ШИМ, для управления интенсивностью свечения диода STAT1. Использовать TIM13. Снять осциллограммы.

4.1. Режим неизменяющейся яркости свечения диода.

4.2. Режим изменяющейся скважности сигнала ШИМ, для плавного увеличения и уменьшения яркости свечения диода, режимы включается при нажатой USER_BUTTON кнопке.

Переключение светодиода с частотой 1КГц, тактирование от внутреннего источника:

Для начала проинициализируем светодиоды и кнопки, вынесем эту инициализацию в отдельную функцию

```
1. void BTN_and_LED_init (void) {
2.     /* For LED init */
3.     GPIO_InitTypeDef    GPIO_InitStruct;
4.     RCC_AHB1PeriphClockCmd(LED_RCC, ENABLE);
5.     GPIO_InitStruct.GPIO_Pin = LED_PIN_1 | LED_PIN_2
6.     | LED_PIN_3 | LED_PIN_4;
7.     GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF;
8.     GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
9.     GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
10.    GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
11.    GPIO_Init(LED_PORT, &GPIO_InitStruct);
12.    /* FOR User Button */
13.    RCC_AHB1PeriphClockCmd(BUTTON_USER_RCC, ENABLE);
14.    GPIO_InitStruct.GPIO_Pin = BUTTON_USER_PIN;
15.    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
16.    GPIO_Init(BUTTON_USER_PORT, &GPIO_InitStruct);
17.    /* FOR WKUP */
18.    RCC_AHB1PeriphClockCmd(BUTTON_WKUP_RCC, ENABLE);
19.    GPIO_InitStruct.GPIO_Pin = BUTTON_WKUP_PIN;
20.    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
21.    GPIO_Init(BUTTON_WKUP_PORT, &GPIO_InitStruct);
22. }
```

Для тактирования от внутреннего источника, необходимо сбросить все настройки системной частоты вызовом функции:

```
RCC_DeInit();
```

После чего нужно, используя предоставленную структуру, настроим таймер TIM6:

```
1. void TIM6_init (void) {
2.     RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE);
3.     TIM_TimeBaseInitTypeDef TIM_INITStruct;
4.     TIM_TimeBaseStructInit(&TIM_INITStruct);
5.     TIM_INITStruct.TIM_Prescaler = 16 - 1;
6.     TIM_INITStruct.TIM_Period = 1000/2;
7.     TIM_TimeBaseInit(TIM6, &TIM_INITStruct);
8.     TIM_ITConfig(TIM6, TIM_IT_Update, ENABLE);
9.     TIM_Cmd(TIM6, ENABLE);
10.    NVIC_EnableIRQ(TIM6_DAC_IRQn);
11. }
```

Теперь настроим обработчик прерываний от таймера 6

```
1. void TIM6_DAC_IRQHandler() {
2.    if (TIM_GetITStatus(TIM6, TIM_IT_Update) != RESET) {
3.      GPIO_ToggleBits(LED_PORT, LED_PIN_1);
4.      TIM_ClearITPendingBit(TIM6, TIM_IT_Update);
5.    }
6. }
```

Снимем показатели с осциллографа:

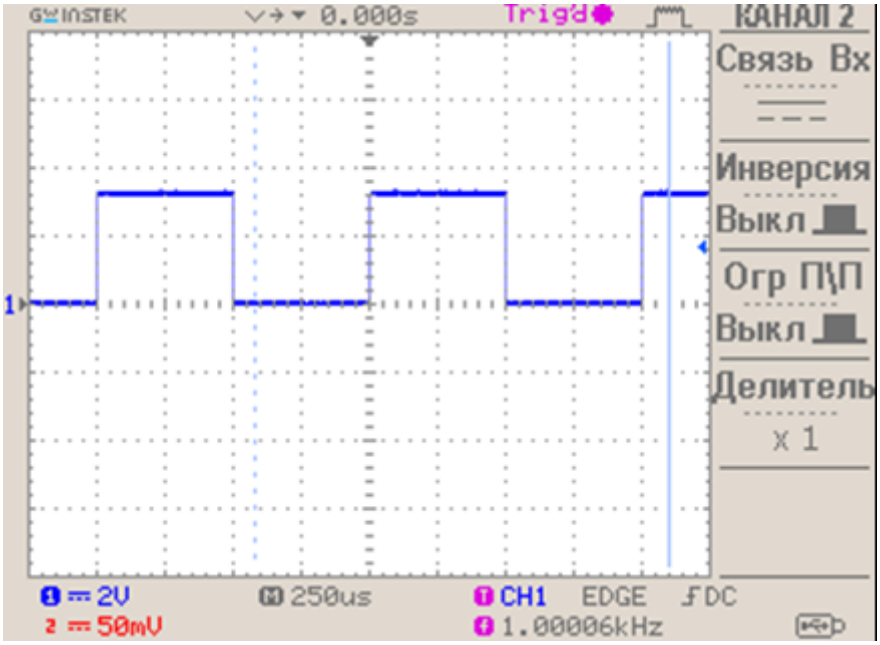


Рисунок 3 – Показания осциллографа при тактировании от внутреннего источника

Переключение светодиода с частотой 1КГц, тактирование от внешнего источника:

В качестве внешнего источника выступает высокоскоростной генератор HSE – 25 МГц. Для начала проинициализируем его:

```
1.  void HSE_init() {
2.      RCC_HSEConfig(RCC_HSE_ON);
3.          if (RCC_WaitForHSEStartUp() == ERROR) {
4.              return;
5.          }
6.      RCC_SYSCLKConfig(RCC_SYSCLKSource_HSE);
7.  }
```

Так как системной частотой теперь является сигнал в 25 МГц, то необходимо изменить настройки таймера, чтобы светодиод переключался по-прежнему с частотой 1КГц. Для этого необходимо изменить настройки делителя в таймере с 16-1 на 25-1 (5 строчка листинга инициализации TIM6).

Снимем показания с осциллографа:

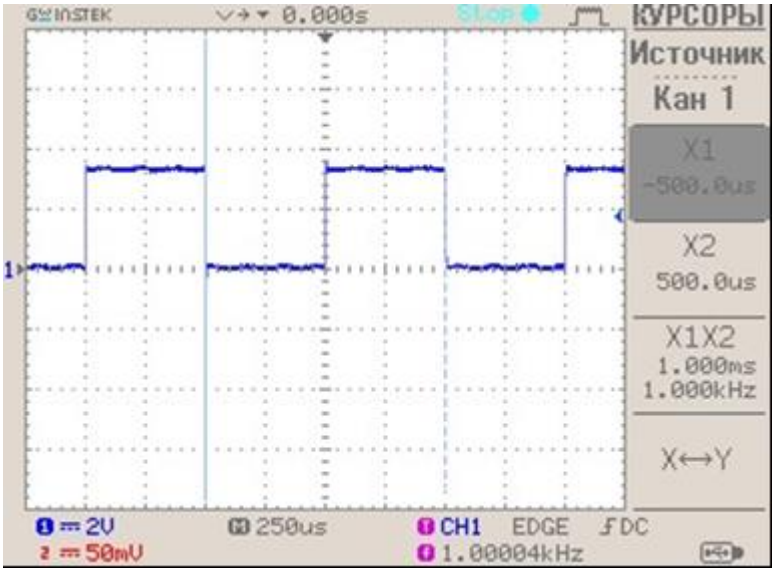


Рисунок 4 – Показания осциллографа при тактировании от внешнего источника

Переключение светодиода с частотой 1КГц, тактирование от PLL:

Для начала затактируем систему от PLL, затем изменим настройки таймера. Приведем следующий листинг текста программы, которая демонстрирует инициализацию PLL. Для начала необходимо высчитать параметр PLL_VCO, который считается по следующей формуле:

$$PLL\_VCO = \left(\frac{[HSE\_VALUE\ or\ HSI\_VALUE]}{PLL\_VCO}\right) * PLL\_NSYCLK = \frac{PLL\_VCO}{PLL\_P}$$

где PLL_M, PLL_N и PLL_P параметры блока PLL. Для получения частоты 168 МГц требуется задать следующие параметры:

PLL_M = 25

PLL_N = 336

PLL_P = 2

PLL_Q = 5

```
1.  void PLL_init () {
2.    // The condition interrupts the function of setting the system fre
3.    quency, if the external generator setting fails*/
4.    RCC_HSEConfig(RCC_HSE_ON);
5.    if (RCC_WaitForHSEStartUp() == ERROR) {
6.       return;
7.    }
8.    // PLL configuration
9.    RCC_PLLConfig(RCC_PLLSource_HSE, 25, 336, 2, 5);
10.   // Enable PLL
11.   RCC_PLLCmd(ENABLE);
12.   RCC_WaitForPLLStartUp();
13.   // Set the bus frequency AHB1 and APB1
14.   RCC_HCLKConfig(RCC_SYSCLK_Div2);
15.   RCC_PCLK1Config(RCC_HCLK_Div1);
16.   RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
17. }
```

Теперь необходимо добавить в программу следующую функцию, которая будет ожидать выход PLL на номинальную частоту:

```
1. void RCC_WaitForPLLStartUp() {
2.    while((RCC - > CR & RCC_CR_PLLRDY) == 0 ) {
3.       __NOP();
4.    }
5. }
```

Снимем показания с осциллографа:

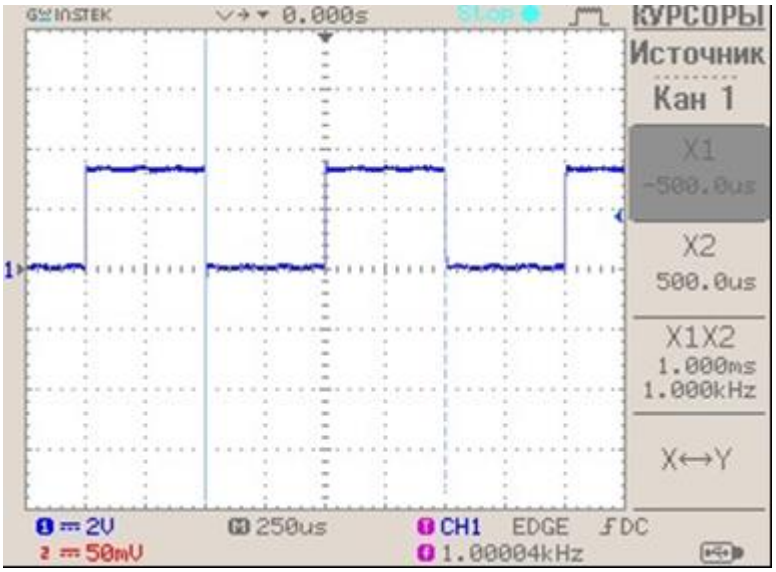


Рисунок 5 – Показания осциллографа при тактировании от PLL

Реализация программы генерации ШИМ сигнала при помощи таймера 13

Для реализации программы генерации ШИМ сигнала необходимо использовать другой таймер. В нашем случае это будет таймер 13. Инициализируем его:

```
    /* For PWM */
    GPIO_PinAFConfig(GPIOF,GPIO_PinSource8,GPIO_AF_TIM13);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM13, ENABLE);
    // Block initialization structures for the configuration of the timer
    // interrupt controller and timer output channel
    TIM_TimeBaseInitTypeDef TIM_INITStruct;
    TIM_OCInitTypeDef TIM_OUTStruct;
    NVIC_InitTypeDef NVIC_INITStruct;
    TIM_TimeBaseStructInit(&TIM_INITStruct);
    TIM_INITStruct.TIM_Prescaler = 0;
    TIM_INITStruct.TIM_Period = 65535;
    // Configuration of the counter, for an ascending account
    TIM_INITStruct.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM13, &TIM_INITStruct);
    // Block initialization fields to configure the controller
    // interrupt
    // Enable overflow interrupt from the 13th timer
    NVIC_INITStruct.NVIC_IRQChannel = TIM1_UP_TIM13_IRQn;
    NVIC_INITStruct.NVIC_IRQChannelPreemptionPrioritty = 0;
    NVIC_INITStruct.NVIC_IRQChannelSubPriority = 0;
    NVIC_INITStruct.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_INITStruct);
    // Channel operation mode - PWM generation
    TIM_OUTStruct.TIM_OCMode = TIM_OCMode_PWM1;
    // Resolution of the timer output channel
    TIM_OUTStruct.TIM_OutputState = TIM_OutputState_Enable;
    // Output value - logical unit
    TIM_OUTStruct.TIM_OCPolarity = TIM_OCPolarity_High;
    // Pulse length (in cycles)
    TIM_OUTStruct.TIM_Pulse = 10000;
    // Initialize timer settings
    TIM_OC1Init(TIM13, &TIM_OUTStruct);
    // Enable interrupt handling by 13 timer
    // overflow
    TIM_ITConfig(TIM13, TIM_IT_Update,ENABLE);
    // Enable Timer work
    TIM_Cmd(TIM13, ENABLE);
```

Теперь необходимо настроить функцию обработчик – прерывания от данного таймера.

```
void TIM8_UP_TIM13_IRQHandler(){
    if (TIM_GetITStatus(TIM13, TIM_IT_Update) != RESET) {
       TIM_ClearITPendingBit(TIM13, TIM_IT_Update);
    }
}
```

Отметим пару пунктов:

1. При генерации ШИМ от таймера стоит учитывать, что ножку МК, на которую будет идти выходной сигнал таймера, необходимо перевести в режим «альтернативной функции».

2. Задание делителя равным нулю и периода равным максимальному значению.

3. Величину пульсации, которая задана как 10000 (в тактах). Изменяется от нуля до значения периода, поэтому следует ее выставить максимальной.

```
1.    GPIO_PinAFConfig(GPIOF,GPIO_PinSource8,GPIO_AF_TIM13);
2.    TIM_INITStruct.TIM_Prescaler = 0;
3.    TIM_INITStruct.TIM_Period = 65535;
4.    TIM_OUTStruct.TIM_Pulse = 10000;
```

Снимем показания с осциллографа:

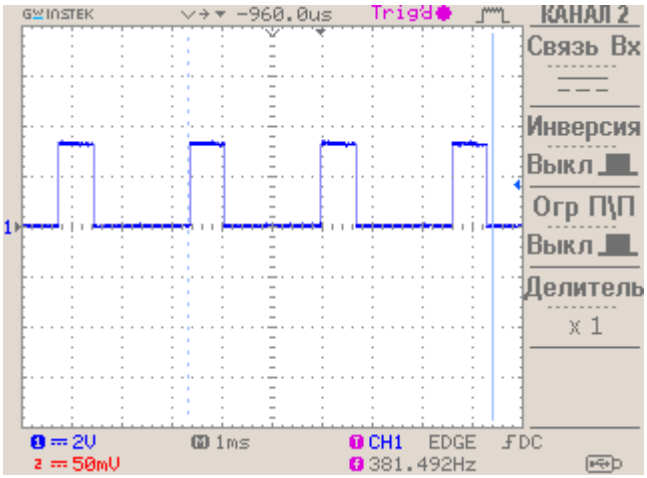


Рисунок 6 – ШИМ-сигнал.

<u>Реализация программы генерации ШИМ сигнала с переменной скважностью:</u>

Суть заключается в следующем: при нажатии кнопки программа должна управлять интенсивностью свечения светодиода и скважностью

ШИМ – сигнала. Задание скважность ШИМ – сигнала происходит при помощи использования функции SPL.

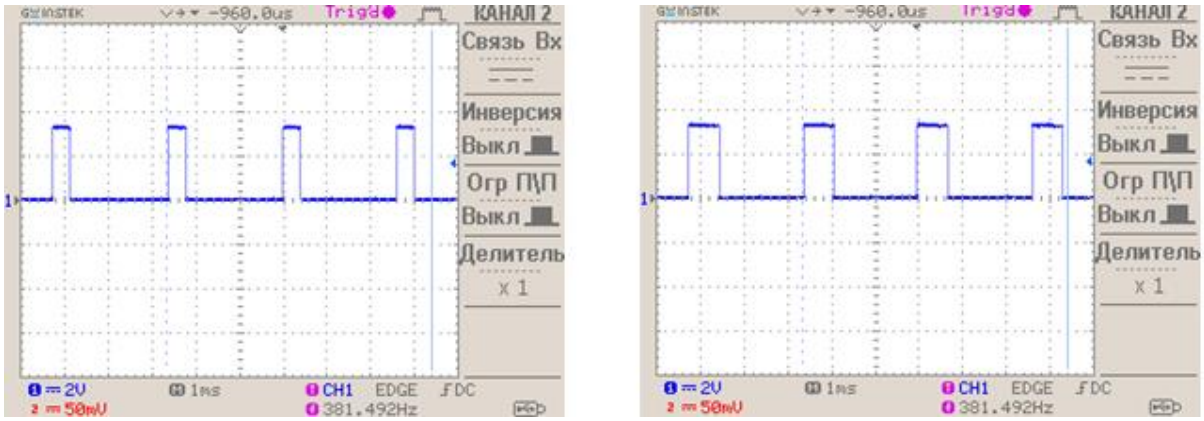
Также необходимо задать стартовое значение скважности:

```
1.    uint32_t TIM_PULSE = 10000;
2.    TIM_SetCompare1(TIM13, TIM_PULSE);
```

Следующий листинг текста программы демонстрирует динамическое изменение скважности ШИМ – сигнала при помощи кнопок WKUP и USER_BTN. В бесконечном цикле происходит считывание кнопок, и, пока они нажаты, меняем значение скважности и переписываем его в таймере. При нажатии USER_BUTTON – уменьшаем скважность, при нажатии WKUP_BUTTON – увеличиваем. В конце чтения кнопки мы ставим функцию задержки, чтобы замедлить чтение кнопки и позволить наблюдать изменение ШИМа.

```
unit8_t readbit = 0x00;
   while(1){
     readbit = GPIO_ReadInputDataBit(BUTTON_USER_PORT,
                                     BUTTON_USER_PIN);
     while (!readbit) {
        if (TIM_PULSE != 0) {
           TIM_PULSE -= 1;
           TIM_SetCompare1(TIM13, TIM_PULSE);
        }
        Delay(DELAY);
     readbit = GPIO_ReadInputDataBit(BUTTON_USER_PORT,
                                      BUTTON_USER_PIN);
     }
     readbit = GPIO_ReadInputDataBit(BUTTON_WKUP_PORT,
                                     BUTTON_WKUP_PIN);
     while (readbit) {
        if (TIM_PULSE != 65535) {
           TIM_PULSE += 1;
           TIM_SetCompare1(TIM13, TIM_PULSE);
        }
        Delay(DELAY);
     readbit = GPIO_ReadInputDataBit(BUTTON_WKUP_PORT,
                                       BUTTON_WKUP_PIN);
   }
```

Снимем показания с осциллографа:



а) б)

Рисунок 7 – Скважность ШИМ-сигнала: а) до нажатия кнопки WKUP_BUTTON; б) при нажатии WKUP_BUTTON

Вывод

В данной лабораторной работе ознакомились с системой тактирования и системными таймерами платы STM32. На практике было реализовали три вида тактирования контроллера от трех различных источников:
PLL, внешний источник и внутренний. Необходимо отметить, что тактирование от PLL отличается от тактирования от внутренних и внешних источников тем, что необходимо правильно высчитать параметр PLL_VCO, который, в свою очередь, зависит от четырех других параметров. Их правильный подбор и является основной особенностью тактирования от PLL.

Также стоит отметить работу по генерации ШИМ – сигнала. Для этого пришлось провести работу по инициализации 13-го таймера, а также отметить несколько особенностей, которые упомянутые в отчете.

Отметим также, особенностью ШИМ сигнала является его изменение скважности, что и было реализовано.