Лабораторная работа №2

ШИМ. Регулятор яркости светодиода. Генератор звуковых колебаний

Цель работы: ознакомиться с возможностями генерации ШИМ с помощью демонстрационной платы; научиться генерировать сигнал с заданными параметрами и использовать его для управления внешними устройствами.

Оборудование и программное обеспечение: плата UDK32F107V, среда разработки Cube IDE, светодиоды – 2шт, резисторы 220 Ом – 2шт, пьезодинамик, беспаечная макетная плата, соединительные провода.

Теоретическая информация

Широтно-импульсная модуляция (ШИМ) – способ управления средним значением напряжения на нагрузке путем изменения скважности импульсов, управляемых ключом. В основном, микроконтроллеры позволяют генерировать цифровой ШИМ различной частоты. Поскольку вывод сигнала ШИМ не является основной функцией пинов, которые подключены к светодиодам, то необходимо выполнить соответствующую их конфигурацию. Для этого следует задать выполнение пинами альтернативных функций. Генерация ШИМ в них связана с использованием дополнительных режимов таймера. Перед подключением устройства известны такие параметры ШИМ, как частота и коэффициент заполнения. Для их расчета в контроллерах STM32 необходимо определить значение предделителя и автоматически загружаемое значение в регистре ARR (Auto-Reload Register). Расчет значения, которое следует записать в предделитель выполняется следующим образом:

$$PSC = \frac{TIMxCLK}{TIMxCNT} - 1,$$

где PSC – значение предделителя, TIMxCLK – входная частота работы таймера, TIMxCNT – частота счетчика.

Для получения необходимой выходной частоты следует записать значений в регистр ARR, которое поучается из следующего соотношения:

$$ARR\_VAL = \frac{TIMxCNT}{TIMx\_out\_freq} - 1,$$

где ARR_VAL – значение для записи в регистр ARR, TIMxCNT – частота счетчика, TIMx_ out_ freq – нужная выходная частота ШИМ.

Последним этапом является задание нужного коэффициента запонения, что обеспечит нужное значени напряжения на выходе. Данная настройка производится с помощью регистра захвата/сравнения (capture/compare register, CCRx), исходя из следующего соотношения:

$$D = \frac{CCRx\_VAL}{ARR\_VAL} * 100\%,$$

где D – коеффициент заполнения, CCRx _VAL – значение в регистре CCRx (x – номер регистра для конкретной линии), ARR_VAL – значение для записи в регистр ARR. Особенностью данных микроконтроллеров является то, что в предделитель и другие регистры можно записать любое значение, которое можно описать с помошью отведенного количества разрядов. Выдача сигнала ШИМ на выход не является основным режимом работы выводов порта, а относится к дополнительным (альтернативным) режимам. Поэтому предварительно нужно задать нужный режим в настройках порта.

1. Регулировка яркости светодиодов при помощи ШИМ

Для начала выполнения работы запускаем Cube IDE и создаём новый проект. Основываясь на опыте из лабораторной работы №1 конфигурируем проект.

Светодиоды требуется подключить к PA2 и PA3 (рис 1). Во избежание выхода из строя портов GPIO, светодиоды требуется подключить через резистор 470 Ом. Кнопки, установленные на плате используют порты PC13 и PA0.

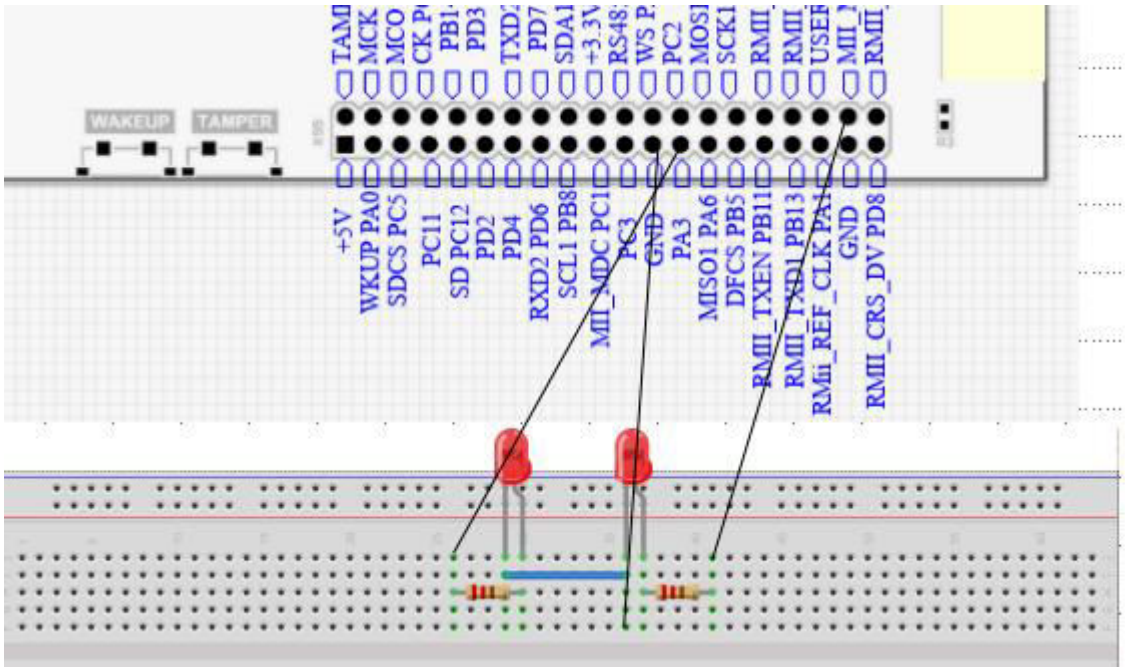

Рис. 1. Собранная схема

Выходы PA2 и PA3 и таймер нужно настроить так, как показано на рис. 2. По окончании конфигурации требуется сохранить проект в папку Lab2.

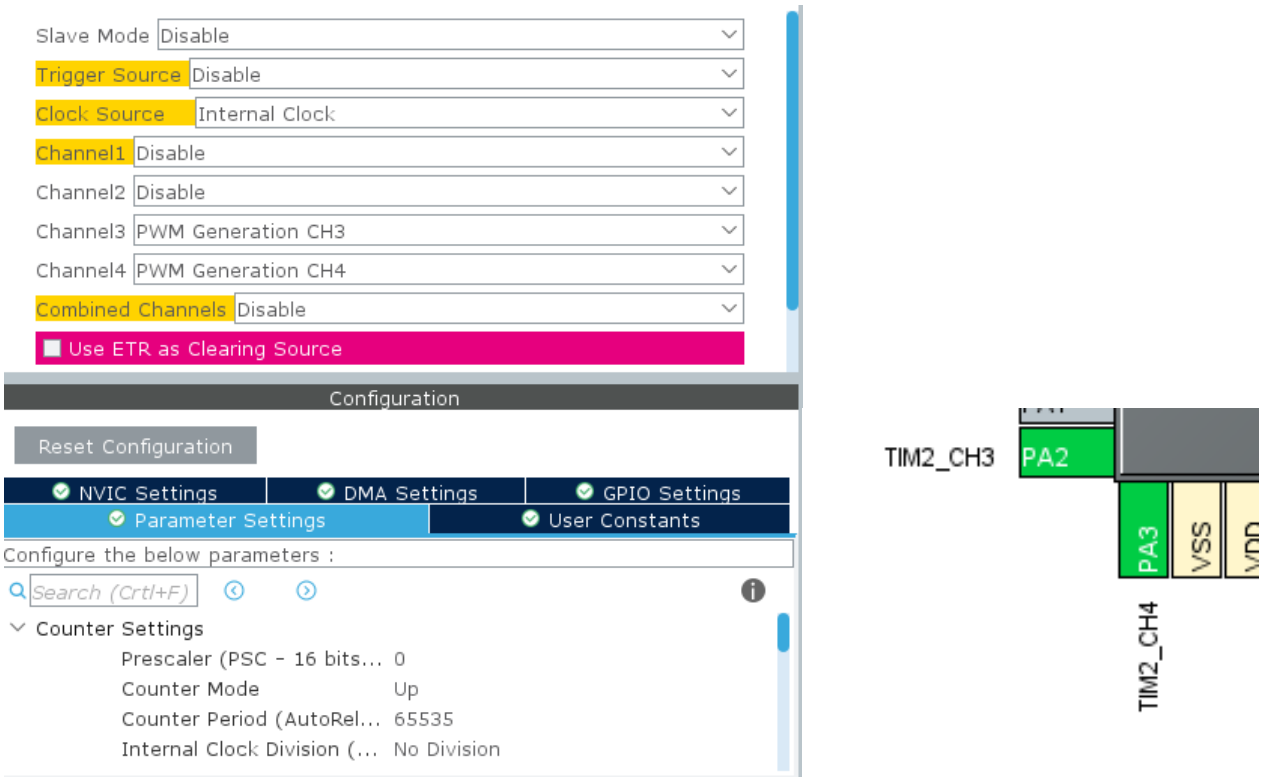

Рис. 2. Конфигурация таймера

Также во вкладке "NVIC Settings" необходимо включить "TIM2 global interrupt".
В private defines нужно прописать:

```
/* USER CODE BEGIN PD */
#define constrain(amt,low,high) ((amt)<(low)?(low):((amt)>(high)?(high):(amt)))
/* USER CODE END PD */
```

Объявляем переменные:

```
  /* USER CODE BEGIN 1 */
    uint32_t i,d;
    uint32_t led_delay = 50;
  /* USER CODE END 1 */
```

Запускаем таймер на каналы 3 и 4 :

```
  /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4);
  /* USER CODE END 2 */
```

Пишем программный код, который будет по очереди плавно включать и плавно выключать светодиоды, а задержка переключения будет регулироваться удержанием кнопок:

```
while (1)
  {
if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){ //если нажата WakeUP, увеличиваем задержку
led_delay -=  50; //декремент
 }

if(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) == 1){ //если нажата Tamper, уменьшаем задержку
```

```
led_delay +=  50; //инкремент
                                    }
for(i=0;i<=262164;i++)
     {
      if(i<65536)        TIM2->CCR3=i;
         else if ((i>65535)&&(i<131072)) TIM2->CCR3=131071-i;
         else if((i>131071)&&(i<196608))        TIM2->CCR4=i-131072;
         else if ((i>196607)&&(i<262164)) TIM2 ->CCR4=262164-i;
         else TIM4->CCR4=262164-i;
      for(d=0;d<led_delay;d++) {
        }

     }
     led_delay = constrain(led_delay, 50, 500); //задаём пределы регулировки
   /* USER CODE END WHILE */
```

В результате увидим, как светодиоды по очереди плавно загораются и гаснут. Удерживая кнопки можно регулировать скорость эффекта.

2. Генерирование звука при помощи ШИМ и пьезодинамика

Помимо регулировки яркости светодиодов, при помощи ШИМ можно также генерировать звук. Прежде, чем начать писать код, нужно изменить конфигурацию таймера, как показано на Рис. 3.

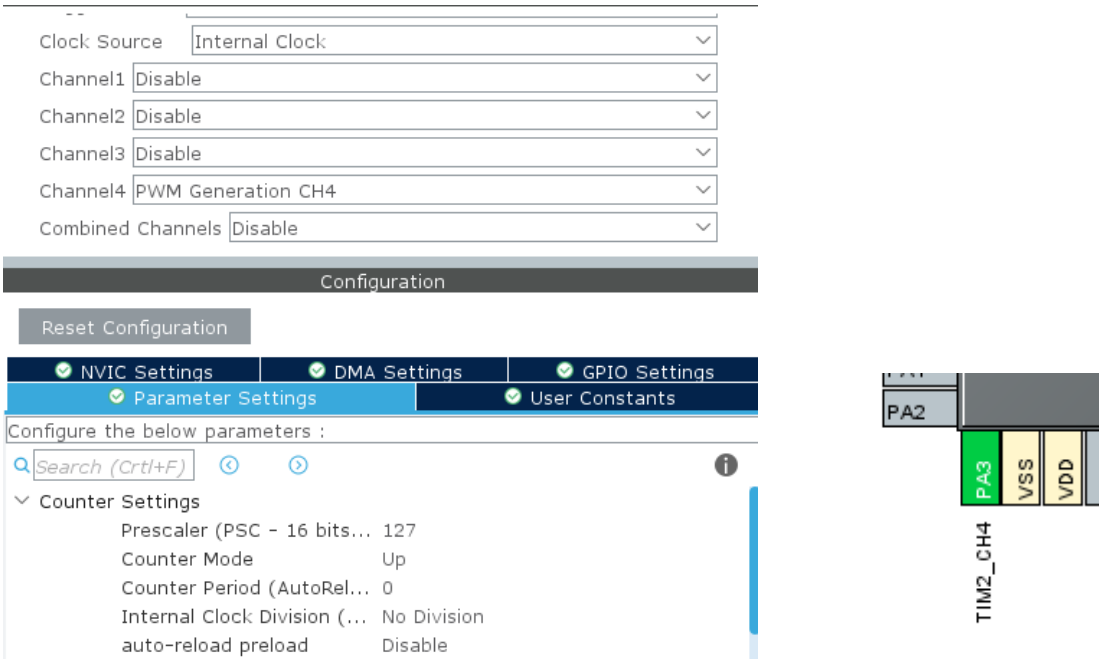

Рис. 3. Конфигурация таймера

По окончании конфигурации приступаем к сборке схемы.

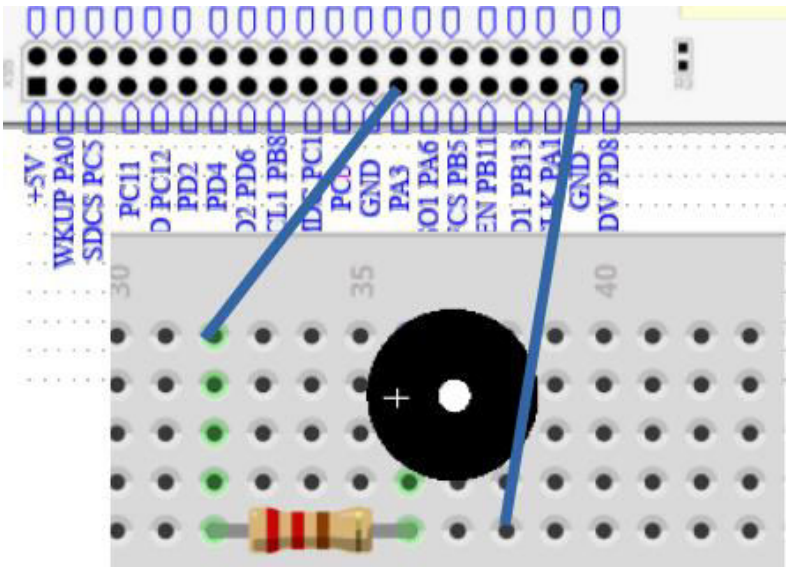

Рис. 4. Схема подключения пьезодинамика

Программный код выглядит следующим образом:

```
 /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4);
/* USER CODE END 2 */



 /* USER CODE BEGIN WHILE */
```

```
 while (1)
  {
          for(int i=600;i<800;i++){
       __HAL_TIM_SET_AUTORELOAD(&htim2, i*2);
       __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_4, i);
            HAL_Delay(5);
            }
            for(int i=800;i>600;i--){
            __HAL_TIM_SET_AUTORELOAD(&htim2, i*2);
            __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_4, i);
            HAL_Delay(5);
            }
    /* USER CODE END WHILE */
```

Прошиваем и если всё правильно работает, то будет слышна сирена. Теперь попробуем воспроизвести мелодию.

Объявляем массивы с мелодией:

```
/* USER CODE BEGIN PV */
int notes[] = { //Ноты
                    392, 392, 392, 311, 466, 392, 311, 466, 392,
                    587, 587, 587, 622, 466, 369, 311, 466, 392,
                    784, 392, 392, 784, 739, 698, 659, 622, 659,
                    415, 554, 523, 493, 466, 440, 466,
                    311, 369, 311, 466, 392};

int times[] = { //Задержка звучания каждой из нот
                    350, 350, 350, 250, 100, 350, 250, 100, 700,
                    350, 350, 350, 250, 100, 350, 250, 100, 700,
                    350, 250, 100, 350, 250, 100, 100, 100, 450,
                    150, 350, 250, 100, 100, 100, 450,
                    150, 350, 250, 100, 750};
```

```
/* USER CODE END PV */
```

В бесконечном цикле прописываем код для воспроизведения:

```
 /* USER CODE BEGIN WHILE */
 while (1)
 {
        for (int i = 0; i < 39; i++)            {
            __HAL_TIM_SET_AUTORELOAD(&htim2, notes[i]*2);
            __HAL_TIM_SET_COMPARE(&htim2,TIM_CHANNEL_4, notes[i]);
            HAL_Delay(times[i]*2);
        }
  /* USER CODE END WHILE */
```

Ход работы

1. На основе кода примера приложения создать свой проект в среде разработки и проверить его работоспособность;
2. Ознакомиться с работой используемых функций, просмотрев исходный код;
3. Реализовать необходимую функциональность;
4. Запрограммировать плату и продемонстрировать работу программы.