МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»

Институт компьютерных наук и технологий

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

Отчет о выполнении лабораторной работы №4 «Использование таймера для формирования заданного временного интервала и аппаратного прерывания для перевода микроконтроллера в режим пониженного энергопотребления» Курс «Программирование микроконтроллеров»

Выполнил студент группы

№5130201/30001 _____ Мелещенко С. И.

Преподаватель _____ Вербова Н. М.

«_____»_____ 2025г.

Санкт-Петербург, 2025

Введение

Тема:

Использование таймера для формирования заданного временного интервала и аппаратного прерывания для перевода микроконтроллера в режим пониженного энергопотребления.

Цель:

Ознакомится с основными методами формирования заданных интервалов времени и перевода микроконтроллера в режим пониженного энергопотребления. Закрепить навыки работы с осциллографом и оценочной платой MCBSTM32F200 в качестве измерительного генератора.

Постановка задачи:

используя библиотеки Keil Vision5, разработать программу для микроконтроллера (МК) STM32F200, которая при помощи таймера формирует периодическое попеременное включение и выключение светодиодов PG6 и PG7 с заданными временными характеристиками (периодом следования переключений и/или длительностью фаз этого периода). При нажатии на кнопку “WAKEUP” программа должна переводить МК в спящий режим, а при ее отпускании пробуждать МК (см. рис. 1).

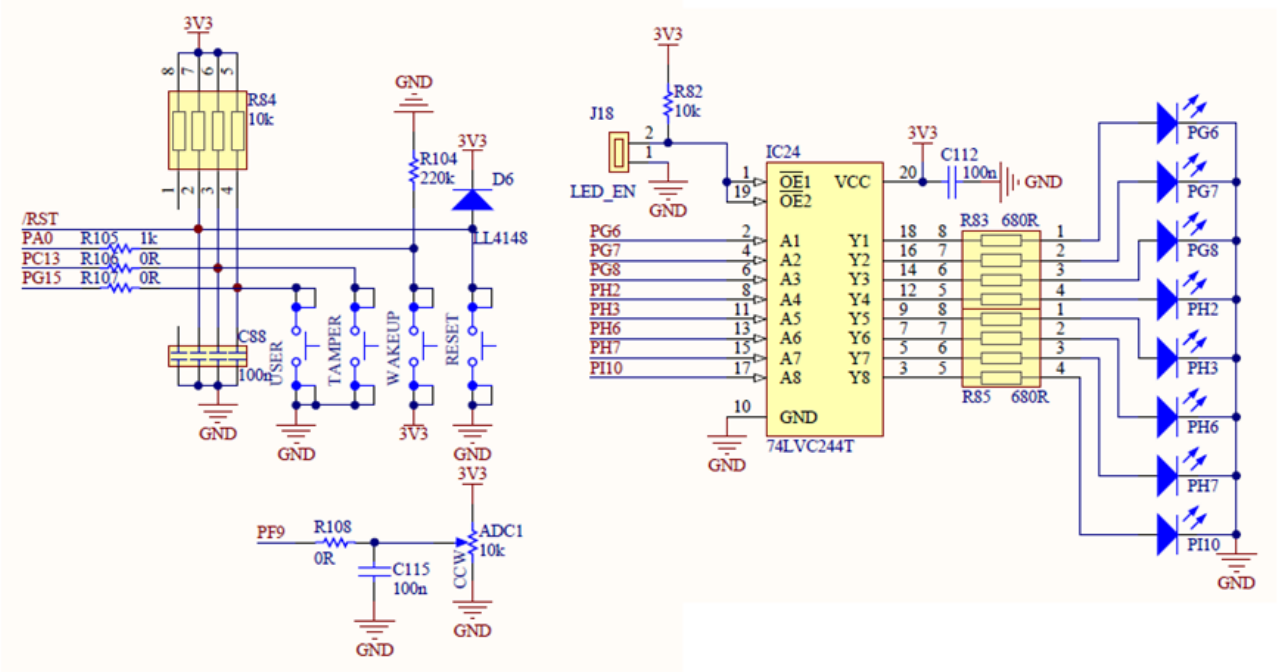


Рис. 1 Схема блоков кнопок и светодиодов

1 Основная часть

1.1 Алгоритм программы

1. Настройка портов PG6 и PG7 как выходов для управления светодиодами.

2. Настройка порта PA0 как вход для обработки нажатия кнопки “WAKEUP”.

3. Инициализация таймера.

4. Настройка прерываний таймера для периодического переключения светодиодов.

5. Настройка линии прерывания EXTI0 для обработки нажатия кнопки “WAKEUP”.

6. Реализация обработчика прерывания для перевода микроконтроллера в спящий режим и пробуждения.

7. Включение и выключение светодиодов PG6 и PG7 с заданным периодом.

8. Обработка нажатия кнопки “WAKEUP” для перевода микроконтроллера в спящий режим и пробуждения.

1.2 Код программы

Общий код программы приведен в приложении.

1. Подключение заголовочных файлов.
include "stm32f2xx.h"
// Подключает стандартные определения и структуры для STM32F2
include "core_cm3.h"
// Подключает ядро Cortex-M3
Эти файлы содержат определения регистров, структуры и макросы, необходимые для работы с микроконтроллером STM32F2.

2. Функция задержки.
void delay ()
unsigned long i;
i = 0;
for (i = 0; i < 2000000; i++)
// Пустой цикл для создания задержки
Программная задержка с помощью цикла.

3. Обработчик прерывания таймера TIM6.
void TIM_DAC_IRQHandler()
if (TIM6->SR & $TIM_S R_U IF$)
// Проверяем флаг обновления (переполнение таймера)
TIM6->SR &= $\tilde{}$TIM_SR_UIF;
// Сбрасываем флаг обновления (обязательно)
GPIOG->ODR || = $1ul$ << 8;
// Включаем светодиод на PG8
delay();
// Ждем
GPIOG->ODR &= $\tilde{}$(1ul « 8);
// Выключаем светодиод
Проверяет, произошло ли прерывание по переполнению таймера TIM6. Если да, сбрасывает флаг и зажигает светодиод на порте PG8 на короткое время.

4. Обработчик прерывания внешнего события (нажатие кнопки).
void EXTI0_IRQHandler(void)
GPIOG->ODR || = $1ul$ << 6;
// Включаем светодиод на PG6
delay();
GPIOG->ODR &= $\tilde{}$(1ul « 6);
// Выключаем светодиод
delay();
EXTI->PR || = $EXTI_PR_PR$0;
// Сбрасываем флаг прерывания по линии 0

Срабатывает при нажатии кнопки (подключенной к PA0). Включает и выключает светодиод на порте PG6. Сбрасывает флаг прерывания для линии EXTI0.

5. Главная функция main.
int main ()
RCC->AHB1ENR $\| = 1ul << 6$;
// Включаем тактирование порта G
RCC->APB2ENR $\| = 1ul << 14$;
// Включаем тактирование SYSCFG
SCB->SCR $\| = 1ul << 2$;
// Переводим процессор в режим глубокого сна (Deep Sleep)
Включает тактирование порта GPIOG. Включает тактирование системного конфигурационного контроллера SYSCFG. Переводит процессор в режим пониженного энергопотребления.

6. Настройка портов ввода-вывода.
GPIOG->MODER = (GPIOG->MODER & $\tilde{}$(1ul « 13)) $\|1ul << 12$;
// Настраиваем PG6 как выход
GPIOG->MODER = (GPIOG->MODER & $\tilde{}$(1ul « 15)) $\|1ul << 14$;
// Настраиваем PG7 как выход
GPIOA->MODER = (GPIOA->MODER & $\tilde{}$(1ul « 1)) & (1ul);
// Настраиваем PA0 как вход
PG6 и PG7 настроены как выходы. PA0 (кнопка) настроен как вход.

7. Настройка прерываний внешнего источника (кнопка).
EXTI->IMR $\| = EXTI\_IMR\_MR0$;
// Разрешаем прерывания по линии 0
EXTI->RTSR $\| = EXTI\_RTSR\_TR0$;
// Прерывание по нарастающему фронту
EXTI->FTSR $\| = EXTI\_FTSR\_TR0$;
// Прерывание по спадающему фронту
SYSCFG->EXTICR[0] $\| = SYSCFG\_EXTICR1\_EXTI0_PA$;
// Привязываем PA0 к линии EXTI0
Активируем прерывания для PA0. Прерывания настроены на оба фронта (нажатие и отпускание кнопки).

8. Настройка приоритета и включение прерываний.
NVIC_SetPriority(6, 5);
// Устанавливаем приоритет прерывания (тип 6, уровень 5)
NVIC_EnableIRQ(6);
// Разрешаем прерывание
Устанавливает приоритет прерывания. Разрешает обработку прерываний.

9. Включение тактирования таймера TIM6.
RCC->APB1ENR $\| = RCC\_APB1ENR\_TIM6EN$;

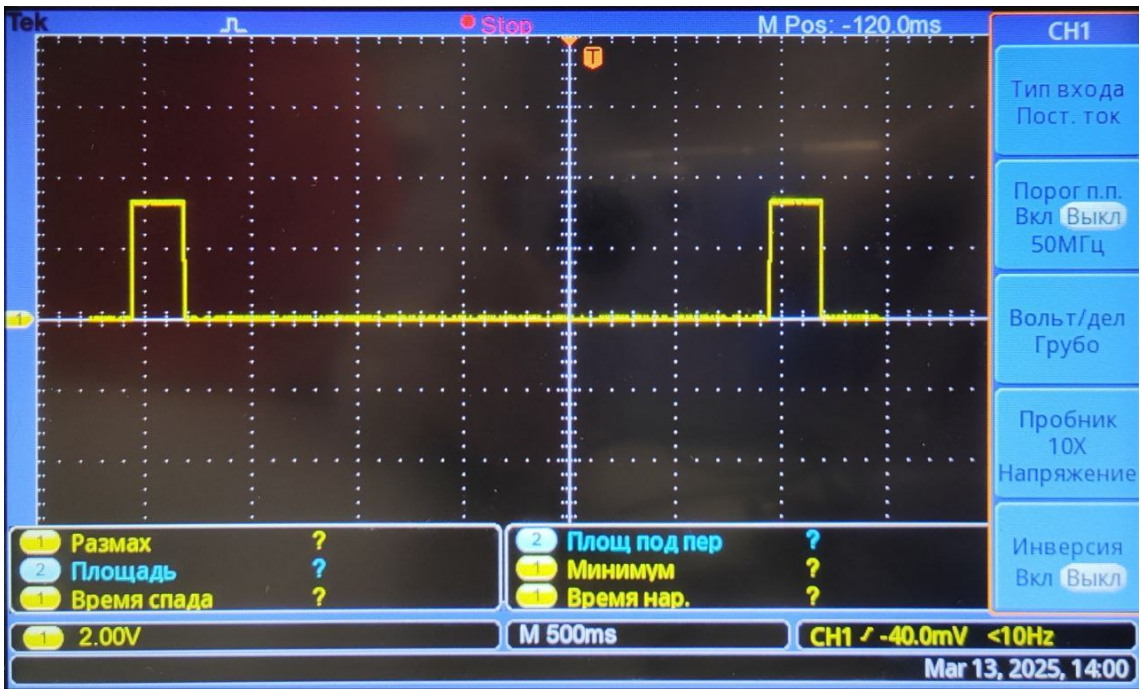
// Включаем тактирование TIM6
Позволяет таймеру работать.

10. Настройка TIM6 для генерации прерываний
TIM6->DIER $\| = TIM\_DIER\_UIE$;
// Разрешаем прерывания по переполнению таймера
TIM6->EGR $\| = TIM\_EGR\_UG$;
// Программно генерируем прерывание (инициализация таймера)
Разрешает генерацию прерываний. Программно запускает обновление таймера.

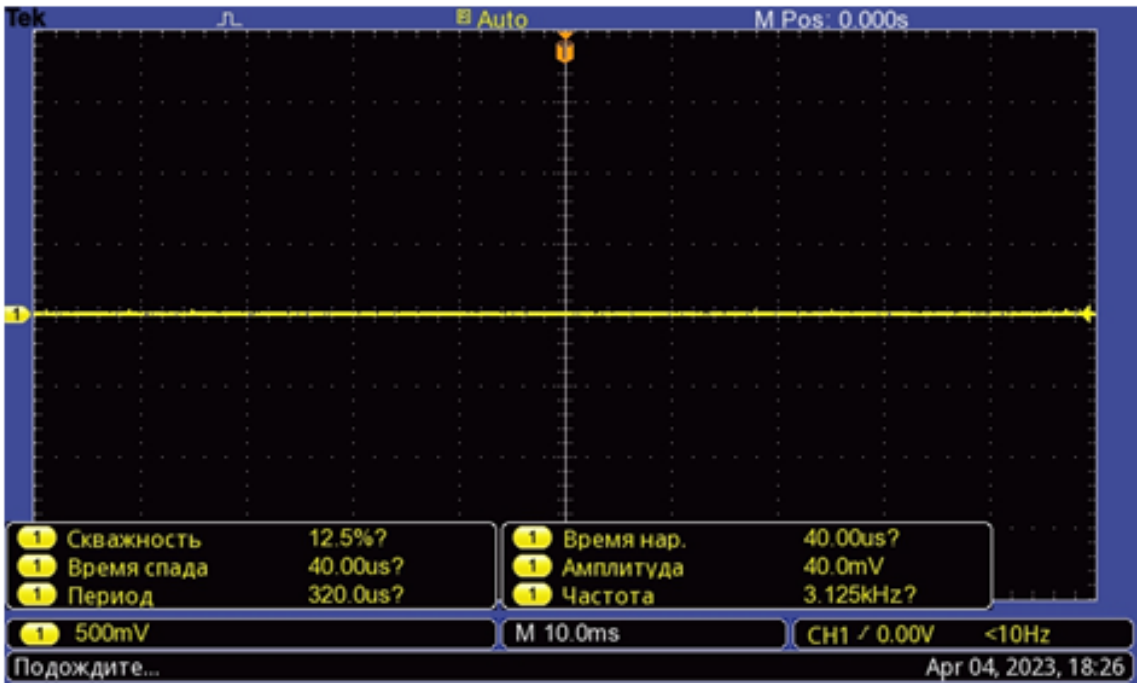
11. Основной цикл работы микроконтроллера.
for (;;)
GPIOG->ODR $\| = 1ul << 7$;
// Включаем светодиод на PG7
delay();
GPIOG->ODR &= $\tilde{1}$ul « 7;
// Выключаем PG7
GPIOG->ODR $\| = 1ul << 6$;
// Включаем PG6
delay();
GPIOG->ODR &= $\tilde{1}$ul « 6;
// Выключаем PG6
Бесконечный цикл. Поочередно мигает светодиодами PG6 и PG7.

1.3 Работа с осцилографом

К плате был подключен осцилограф для измерения временной информации сигнала с помощью встроенных функций.



Програмное измерение периода, частоты сигнала, ширины положительного и отрицательного уровней (состояние “включено”/состояние “выключено”), коэффициента заполнения периода (скважности) представлено на изображении ниже.



1.4 Ответы на вопросы

1. Какова частота следующего сигнала Y= 100 Cos(500**t) где Y принимает амплитудное значение в момент времени t=0.

[C] 250

2. Какова частота следующего сигнала Y= 100 Cos(500**t) + 200 Sin(500**t) где Y принимает амплитудное значение в момент времени t=0. [C] 250

3. Каков период сигнала представленного в вопросе 2

[C] 1/ 250

4. Каково значение времени нарастания сигнала если сигнал достигает 10%, 20%, 80%, и 90% своего амплитудного значения в моменты времени соответственно 2 миллисекунды, 2.02 миллисекунды, 2.08 миллисекунды, и 2.12 миллисекунды соответственно?

[A] 0.12 миллисекунды

5. Какова разность фаз между следующими сигналами Y= 100 Cos(500**t + 30) and Y= 100 Sin(500**t)

[D] 120

1.5 Постлабораторное оценивание

1. Перечислите концепции, которые Вы изучили, выполняя это упражнение:

Работа с таймерами для формирования временных интервалов, реализация аппаратных прерываний для управления режимами энергопотребления. использование осциллографа для измерения временных характеристик сигналов.

2. Перечислите встроенные в осциллограф функции для измерения временной информации:

Измерение периода и частоты сигнала, времени нарастания и спада сигнала, времени высокого и низкого уровней ШИМ сигнала, фазового сдвига и времени запаздывания между двумя сигналами.

3. Перечислите 8 экспериментов предписанных учебным планом, где Вы можете применить концепции измерений, изученные в данном эксперименте:

Исследование работы таймеров микроконтроллера, измерение временных характеристик сигналов, реализация ШИМ-модуляции, исследование режимов энергопотребления микроконтроллера, работа с прерываниями, измерение фазового сдвига между сигналами, исследование работы светодиодов и кнопок, использование осциллографа для анализа сигналов.

4. Сгенерируйте с помощью MCBSTM32F200 прямоугольную волну и измерьте значения времени состояния “включено”, состояния “выключено”, периода и коэффициента заполнения, используя:

Результаты, полученные обоими методами, совпадают с небольшой погрешностью.

2 Вывод

В ходе лабораторной работы был использован таймер для формирования заданных временных интервалов, реализовано аппаратное прерывание для перевода микроконтроллера в режим пониженного энергопотребления, освоена работа с осциллографом для измерения временных характеристик сигналов.

Программа корректно управляет светодиодами, обеспечивая их попеременное включение и выключение. Режим пониженного энергопотребления работает стабильно, что подтверждается измерениями с помощью осциллографа. Учтены особенности дребезга контактов кнопки, что позволило избежать ложных срабатываний прерываний.

Приложение

Листинг 1: Код программы

```
#include "stm32f2xx.h"
#include "core_cm3.h"
#include "core_cm0.h"

void delay ()
{
unsigned long i;
i=0;
for(i=0; i<2000000; i++){}
}

void TIM_DAC_IRQHandler()
{
        if(TIM6->SR&TIM_SR_UIF)
        {
                TIM6->SR&=~TIM_SR_UIF;
                GPIOG->ODR |= 1ul<<8;
                delay ();
                GPIOG->ODR &= ~(1ul<<8);
        }
}
void EXTI0_IRQHandler(void)
{
        GPIOG->ODR |= 1ul<<6;
        delay ();
        GPIOG->ODR &= ~1ul<<6;
        delay ();
        EXTI->PR|=EXTI_PR_PR0;
}
int main ()
{
        RCC->AHB1ENR |= 1ul<<6;
        RCC->APB2ENR|= 1ul<<14;
        SCB->SCR |= 1ul<<2;

        GPIOG->MODER = (GPIOG->MODER & ~(1ul<<13)) | 1ul<<12;//PG6
        GPIOG->MODER = (GPIOG->MODER & ~(1ul<<15)) | 1ul<<14;//PG7
        GPIOA->MODER = (GPIOA->MODER & ~(1ul<<1)) & ~(1ul);//PA0

        EXTI->IMR|=EXTI_IMR_MR0;
        EXTI->RTSR|= EXTI_RTSR_TR0;
        EXTI->FTSR|= EXTI_FTSR_TR0;
        SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA;
        NVIC_SetPriority(6,5);
        NVIC_EnableIRQ(6);
        RCC->APB1ENR|=RCC_APB1ENR_TIM6EN;

        TIM6->DIER|=TIM_DIER_UIE;

        TIM6->EGR|=TIM_EGR_UG;


for (;;)
{
```

```
GPIOG->ODR |= 1ul<<7;
delay ();
GPIOG->ODR &= ~1ul<<7;
GPIOG->ODR |= 1ul<<6;
delay ();
GPIOG->ODR &= ~1ul<<6;
}
```