

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 №4.

Работу выполнил:

Путята М.А.

студент группы 5130201/30002

Проверила:

Вербова Н. М.

Санкт-Петербург - 2025 г.

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 μ 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).

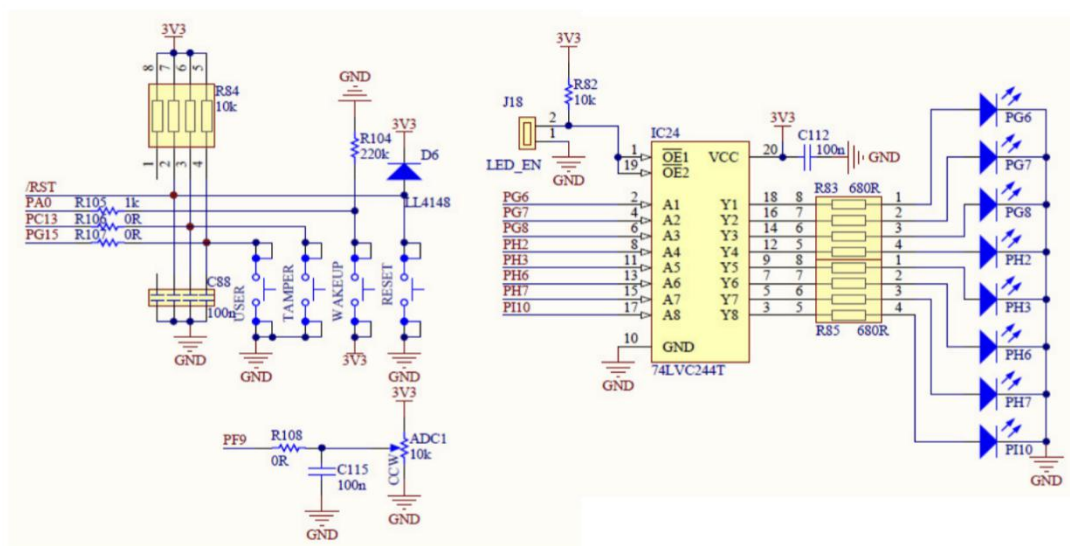


Рис. 1 Схема блоков кнопок и светодиодов

Теоретические данные:

Для формирования интервалов времени (событий) а также для измерения этих интервалов в МК используется аппаратная подсистема – таймер.

Таймер – это устройство, автоматически устанавливающее время начала и конца какого-либо процесса или отдельных его этапов.

МК STM32F200 оснащен 8 периферийными программируемыми 16 разрядными таймерами TIM1 – TIM8 состоящими из перезагружаемых счетчиков управляемых

программируемыми устройствами предварительного масштабирования. Они могут быть использованы для различных целей, включая измерение длительности импульсов входных сигналов или генерирования выходных прямоугольных импульсов, например ШИМ с вставкой “мертвой” зоны. При использовании устройствами предварительного масштабирования и контроллера тактового сигнала RCC длительность импульсов и период их следования могут быть изменены от нескольких микросекунд до нескольких миллисекунд. Таймеры полностью независимы и не потребляют никаких ресурсов. Они могут быть также синхронизированы совместно.

В ядре микроконтроллера находится еще один таймер – системный таймер SysTick (см. файл CD00228163.pdf). Этот таймер предназначен для формирования временных интервалов операционной системы реального времени – RTOS. Однако периодические прерывания, формируемые этим таймером можно использовать и для других целей. В микроконтроллерах на ядре Cortex время перехода к обработчику прерывания строго определено, что является огромным плюсом этого ядра.

Как только регистры таймера будут загружены необходимыми значениями, будет разрешена генерация прерываний и запущен счет,

контроллер прерываний NVIC начинает ожидать от данного таймера прерывания и в МК появится источник периодических прерываний, при этом периферийные таймеры могут быть использованы для других целей.

Для конфигурирования таймера в файле `core_cm3.h` создана функция `__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)`, в качестве аргумента которой передается коэффициент деления тактовой частоты для получения необходимой временной задержки. По умолчанию тактовая частота ядра равна 12 МГц.

Выработка тактового сигнала высокопроизводительного 32 разрядного процессора Cortex-M3 останавливается при переходе его в режимы со сверхнизким потреблением энергии. Это режимы сна и глубокого сна. В режиме глубокого сна дополнительно отключаются флэш-память и основной тактовый генератор. Для управления этими режимами используется регистр управления системой SCB_SCR (см. файл CD00228163.pdf). При его конфигурировании под режим сна используется конструкция `SCB_SCR_SLEEPONEXIT_Msk` файла `core_cm3.h`.

Код программы:

```
#include "stm32f2xx.h"           // Device header

#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; // Enable port G clocking

RCC->APB2ENR|= 1ul<<14; //SYSCFGEN

SCB->SCR |= 1ul<<2; //перевели в deepsleed

```

```

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; //зарезервировали две линии под
прерывания (сконфигурировали маскирующие биты...)

EXTI->RTSR|= EXTI_RTSTR_TR0; //Rise Signal

EXTI->FTSR|= EXTI_FTSR_TR0; //Fall Signal

```

```
SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA;
```

```
//прикрепили pa0 к зарезарвированной линии
```

```
NVIC_SetPriority(6,5); //(номер в таблице векторов прерываний(тип),  
приоритет)
```

```
NVIC_EnableIRQ(6); //активировали прерывание
```

```
RCC->APB1ENR|=RCC_APB1ENR_TIM6EN;//включили тактовый  
сигнал для tim6
```

```
// Разрешаем таймеру генерацию прерываний
```

```
// (по умолчанию, после сброса бит TIM_CR1_URS сброшен в 0
```

```
// и прерывание генерируется как при переполнении счётчика,
```

```
// так и при установке бита TIM_EGR_UG).
```

```
TIM6->DIER|=TIM_DIER_UIE;
```

```
// При выполнении следующей строки генерируется прерывание
```

```
// (при этом сам таймер пока ещё остановлен: бит включения
```

```
// счёта TIM_CR1_CEN сброшен в 0).
```

```
TIM6->EGR|=TIM_EGR_UG;
```

```
for (;;) 
```

```
{
```

```
GPIOG->ODR |= 1ul<<7;
```

```
delay ();
```

```
GPIOG->ODR &= ~1ul<<7;
```

```
GPIOG->ODR |= 1ul<<6;
```

```
delay ();
```

```
GPIOG->ODR &= ~1ul<<6;
```

```
}
```

```
}
```

Алгоритм программы:

Сперва стандартно подключаются несколько библиотек. Стандартно определяется функция delay().

Переопределяется функция обработки прерываний от стандартного таймера TIM и DAC (ЦАП – цифрово аналоговый преобразователь).

Обработчик прерывания от таймера должен сбросить бит TIM_SR_UIF регистра TIMx->SR путём записи в бит 0. Иначе, после возврата из прерывания, будет снова вызван его обработчик.

Проверяем, было ли прерывание от таймера в условии. Если да, то

сбрасываем бит TIM_SR_UIF. Светодиод G8 на время загорается, затем гаснет.

Переопределяется прерывание для нулевой линии. При прерывании по нажатию кнопки WAKEUP светодиод G6 на время загорается, затем гаснет. И сбрасывается флаг прерывания.

В основном теле программы настраиваем выходы PG6 и PG7 на вывод цифровых данных аналогично предыдущим лабораторным работам. Переводим МК в состояние DeepSleep, чтобы избежать дребезг сигнала. Резервируем две линии под прерывания (skonфигурировали маскирующие биты), настраиваем линию прерывания на возрастающий и падающий фронт. Прикрепляем pa0 к зарезарвированной линии.

Устанавливаем приоритет прерывания с помощью регистров приоритета прерывания NVIC_IPR (см. файл CD00228163.pdf). Для линии использовалась функция “void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority))” из строки 1464 файла core_cm3.h. В данной функции на первом месте стоит тип прерывания, то есть его позиция в таблице прерываний, а на втором непосредственно приоритет. И активируем прерывание. Для этого необходимо воспользоваться

справочным руководством по программированию (см. файл CD00228163.pdf) активация обработки определенного вектора прерывания осуществляется с помощью регистров NVIC_ISER. Для линии использовалась функция “void NVIC_EnableIRQ(IRQn_Type IRQn)” из строки 1382 файла core_cm3.h.

Разрешаем таймеру генерацию прерываний (по умолчанию, после сброса бит TIM_CR1_URS сброшен в 0 и прерывание генерируется как при переполнении счётчика, так и при установке бита TIM_EGR_UG).

Запускаем цикл в котором поочередно запускаются и гаснут светодиоды G6 и G7 (Рис.2).

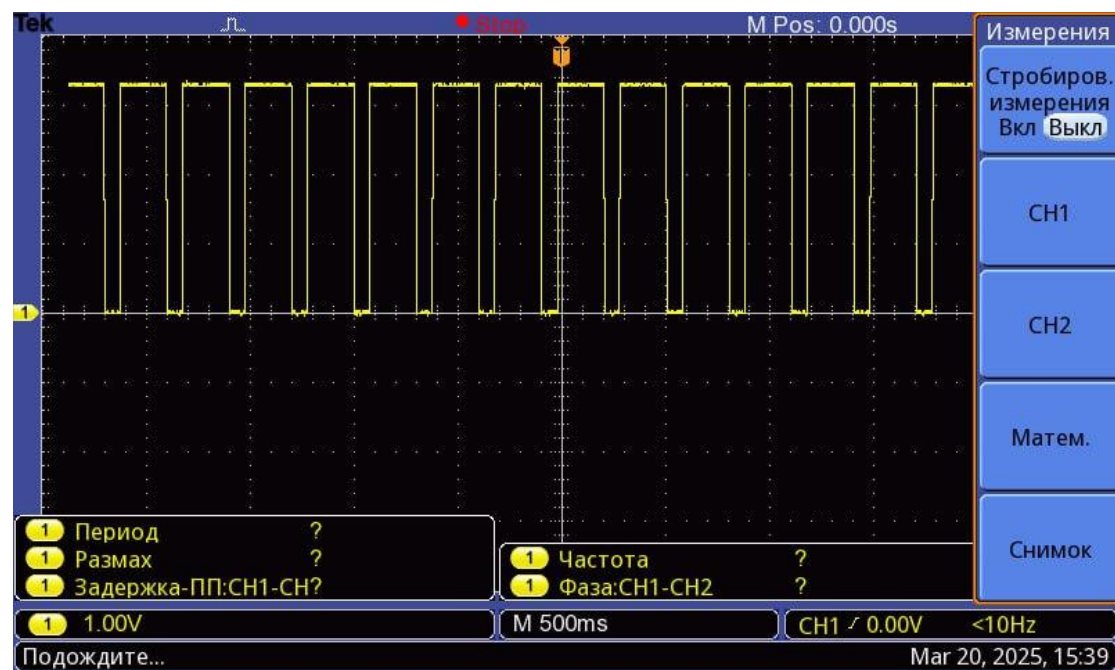


Рис.2

Работа с осциллографом:

Програмное измерение периода, частоты сигнала, ширины положительного и отрицательного уровней (состояние “включено”/состояние “выключено”) (Рис. 3, 4), коэффициента заполнения периода (скважности) представлено на изображении ниже (Рис.5).

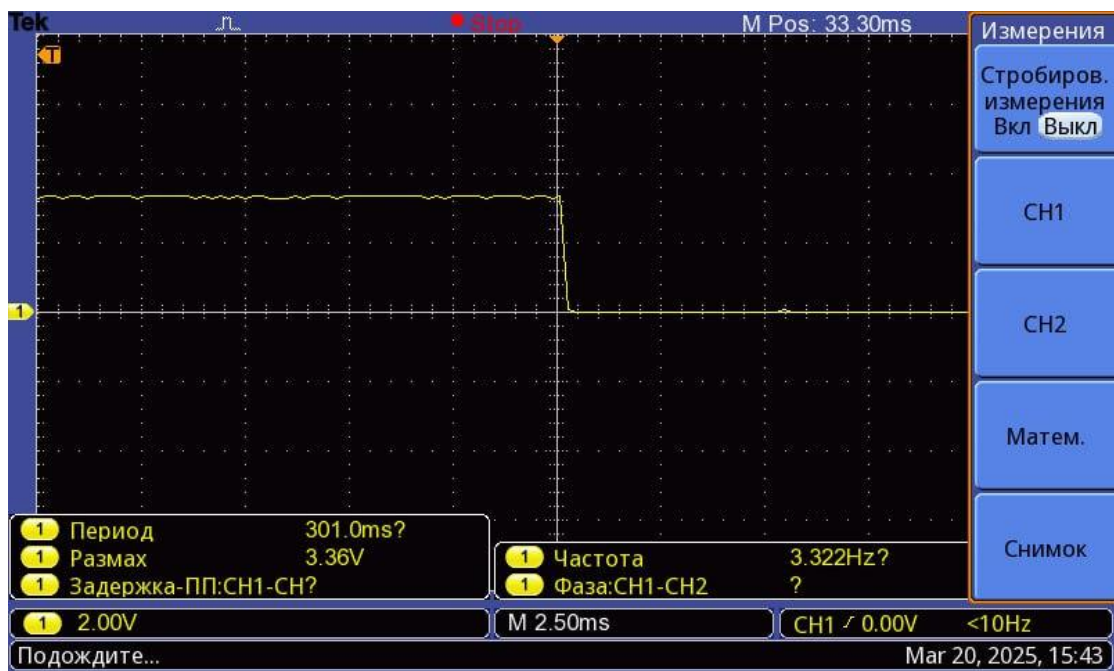


Рис.3

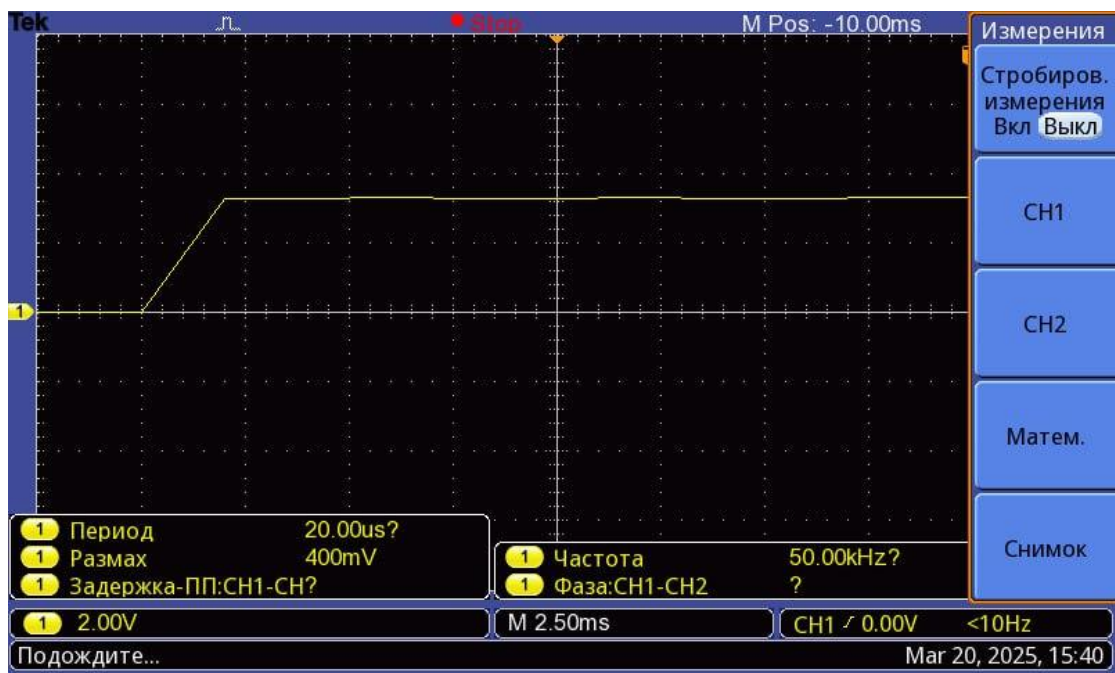


Рис. 4

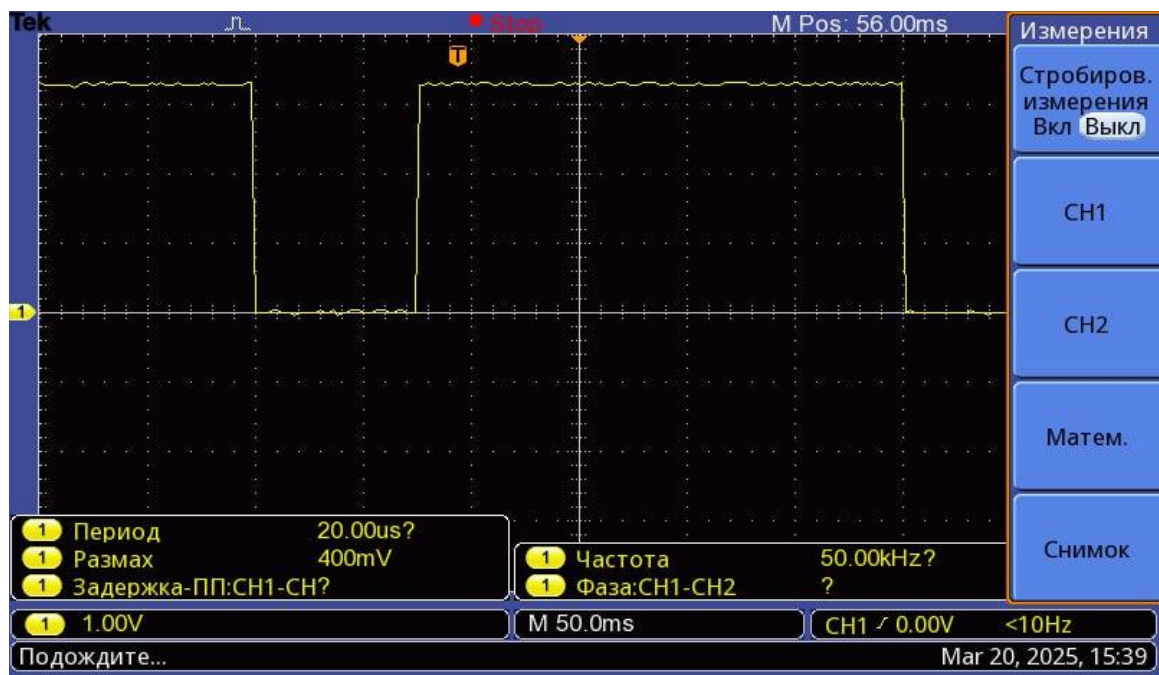


Рис.5

Вывод:

Были освоены основные методы формирования заданных интервалов времени и перевода микроконтроллера в режим пониженного энергопотребления. Закреплены навыки работы с осциллографом и оценочной платой MCBSTM32F200 в качестве измерительного генератора.