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 №3
«События и прерывания»
Курс «Программирование микроконтроллеров»

Выполнил студент группы

№5130201/30001 _____ Мелещенко С. И.

Преподаватель _____ Вербова Н. М.

«_____»_____ 2025г.

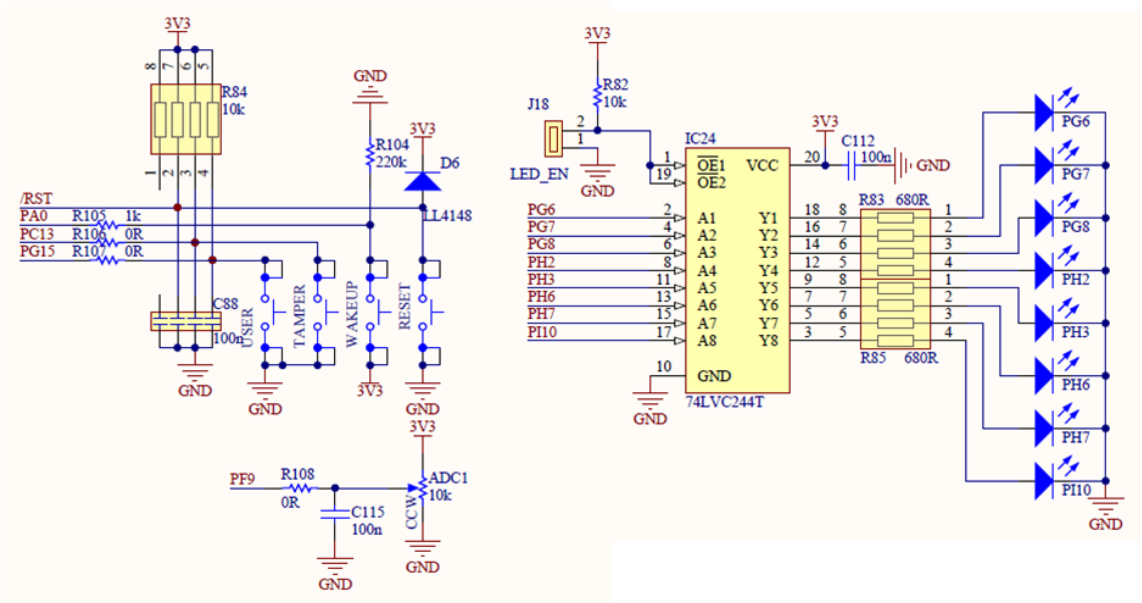
Санкт-Петербург, 2025

Введение

Тема: События и прерывания.

Цель: Ознакомится с основными методами обработки событий и прерываний.

Постановка задачи: разработать программу для микроконтроллера (МК) STM32F200 мигающую светодиодом PG7 и регистрирующую и обрабатывающую с разным приоритетом замыкание кнопок “WAKEUP” и “USER”. При нажатии кнопки “WAKEUP” на некоторое время должен подключаться светодиод PG6, а при нажатии кнопки “USER” светодиод PG8.



1 Основная часть

1.1 Алгоритм программы

1. Настройка портов GPIO.

   Настройка выводов PG6, PG7 и PG8 на вывод цифровых данных.
   Настройка выводов PA0 и PG15 на ввод цифровых данных.

2. Подключение тактирования контроллера конфигурации системы SYSCFG.

3. Настройка контроллера внешних событий/прерываний EXTI.

   Конфигурирование линий прерываний для кнопок "WAKEUP"и "USER".
   Настройка регистров EXTI_IMR, EXTI_RTSR и EXTI_FTSR для выбора фронтов прерываний.
   Настройка регистров SYSCFG_EXTICR для подключения линий EXTI к соответствующим портам.

4. Настройка NVIC.

   Установка приоритетов прерываний с помощью регистров NVIC_IPR.
   Активация прерываний через регистры NVIC_ISER.

5. Основное тело программы.

   Бесконечный цикл, в котором светодиод PG7 мигает с заданной частотой.
   Обработчики прерываний для кнопок "WAKEUP"и "USER которые включают соответствующие светодиоды на некоторое время.

6. Сборка и запуск программы.

   Сборка проекта в Keil Vision5.
   Запуск отладочного режима и загрузка программы в оценочную плату.

7. Тестирование программы.

   Наблюдение за работой светодиодов при нажатии кнопок.
   Изменение приоритетов прерываний и наблюдение за изменением порядка их обработки.

1.2 Код программы

Общий код программы приведен в приложении 1.

1. include "stm32f2xx.h"// Заголовочный файл для работы с микроконтроллером STM32F2xx

include "core_cm3.h"// Заголовочный файл для работы с ядром Cortex-M3

Подключены необходимые заголовочные файлы для работы с микроконтроллером STM32F2xx и ядром Cortex-M3. Эти файлы содержат определения регистров и функций, используемых в программе.

2. Функция задержки

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

Функция delay создает задержку с помощью пустого цикла. Это простой способ реализовать временную задержку в программе. В данном случае цикл выполняется 2 миллиона раз, что создает заметную задержку.

3. Обработчик прерывания для EXTI0 (кнопка "WAKEUP")

void EXTI0_IRQHandler(void) GPIOG->ODR |= 1ul « 6; // Включаем светодиод PG6

delay(); // Задержка

GPIOG->ODR &= ~1ul « 6; // Выключаем светодиод PG6

delay(); // Задержка

EXTI->PR |= EXTI$_P R_P R$0; //$EXTI$0

Этот обработчик прерывания срабатывает при нажатии кнопки "WAKEUP" (линия EXTI0). Он включает светодиод PG6 на некоторое время, затем выключает его и сбрасывает флаг прерывания, чтобы система могла обрабатывать новые прерывания.

4. Обработчик прерывания для EXTI15 (кнопка "USER")

void EXTI15_10_IRQHandler(void)

GPIOG->ODR |= 1ul « 8; // Включаем светодиод PG8

delay(); // Задержка

GPIOG->ODR &= ~1ul « 8; // Выключаем светодиод PG8

delay(); // Задержка

EXTI->PR |= EXTI$_P R_P R$0; //$EXTI$15

Этот обработчик прерывания срабатывает при нажатии кнопки "USER"(линия EXTI15). Он включает светодиод PG8 на некоторое время, затем выключает его и сбрасывает флаг прерывания.

5. Основная функция main

int main() RCC->AHB1ENR |= 1ul « 6; // Включаем тактирование порта G

RCC->APB2ENR |= 1ul « 14; // Включаем тактирование SYSCFG

GPIOG->MODER = (GPIOG->MODER & ~(1ul « 13))

| 1ul « 12; // PG6 как выход

GPIOG->MODER = (GPIOG->MODER & ~(1ul « 15))

| 1ul « 14; // PG7 как выход
GPIOG->MODER = (GPIOG->MODER & ~(1ul «
17)) | 1ul « 16; // PG8 как выход
GPIOG->MODER = (GPIOG->MODER & ~(1ul «
31)) & ~(1ul « 30); // PG15 как вход
GPIOA->MODER = (GPIOA->MODER & ~(1ul «
1)) & (1ul); // PA0 как вход
EXTI->IMR |= EXTI_IMR$_MR0|$
EXTI_IMR_MR15; // Разрешаем прерывания по линиям EXTI0 и EXTI15
EXTI->RTSR |= EXTI_RTSR_TR0; // Прерывание по нарастающему фронту для EXTI0
EXTI->FTSR |= EXTI_FTSR_TR15; // Прерывание по падающему фронту для EXTI15
SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA; // Подключаем EXTI0 к порту PA0
SYSCFG->EXTICR[3] |= SYSCFG_EXTICR4_EXTI15_PG; // Подключаем EXTI15 к порту PG15
NVIC_SetPriority(6, 5); // Устанавливаем приоритет прерывания EXTI0
NVIC_SetPriority(40, 6); // Устанавливаем приоритет прерывания EXTI15
NVIC_EnableIRQ(6); // Активируем прерывание EXTI0
NVIC_EnableIRQ(40); // Активируем прерывание EXTI15
for (;;)
GPIOG->ODR |= 1ul « 7; // Включаем светодиод PG7
delay(); // Задержка
GPIOG->ODR &= ~1ul « 7; // Выключаем светодиод PG7
delay(); // Задержка

Тактирование: Включается тактирование порта G и SYSCFG.
Настройка портов:
PG6, PG7 и PG8 настраиваются как выходы.
PG15 и PA0 настраиваются как входы.
Настройка прерываний:
Разрешаются прерывания по линиям EXTI0 и EXTI15.
Настраиваются триггеры прерываний: по нарастающему фронту для EXTI0 и по падающему фронту для EXTI15.
Подключаются линии EXTI0 и EXTI15 к соответствующим портам (PA0 и PG15).
Приоритеты прерываний:
Устанавливаются приоритеты для прерываний EXTI0 и EXTI15.
Активация прерываний:
Активируются прерывания EXTI0 и EXTI15.
Основной цикл:
Включается и выключается светодиод PG7 с задержкой.

Прерывания: Программа использует прерывания для обработки нажатий кнопок. Прерывания настроены на разные фронты сигналов (нарастающий и падающий).

Приоритеты: Прерывания имеют разные приоритеты, что позволяет управлять порядком их обработки.

Светодиоды: Светодиоды PG6 и PG8 управляются через прерывания, а PG7 мигает в основном цикле программы.

2 Вывод

Полученные результаты.

Светодиод PG7 мигал с заданной частотой.

При нажатии кнопки "WAKEUP"светодиод PG6 включался на некоторое время.

При нажатии кнопки "USER"светодиод PG8 включался на некоторое время.

Приоритеты прерываний были настроены таким образом, что прерывание от кнопки "WAKEUP"имело более высокий приоритет, чем прерывание от кнопки "USER".

При одновременном нажатии обеих кнопок сначала обрабатывалось прерывание от кнопки "WAKEUP а затем от кнопки "USER".

При изменении приоритетов на противоположные (прерывание от кнопки "USER"стало более приоритетным), порядок обработки изменился: сначала обрабатывалось прерывание от кнопки "USER а затем от кнопки "WAKEUP".

При установке одинаковых приоритетов порядок обработки прерываний определялся порядком их возникновения.

При изменении группировки приоритетов наблюдалось, что прерывания из группы с более высоким приоритетом прерывали обработку прерываний из группы с более низким приоритетом.

Программа успешно реализована с использованием прерываний и событий на микроконтроллере STM32F200.

Настройка портов GPIO, контроллера EXTI и NVIC выполнена в соответствии с требованиями.

Прерывания успешно обрабатываются в зависимости от их приоритета.

Вложенность прерываний и их приоритеты позволяют гибко управлять порядком обработки событий.

Изменение приоритетов прерываний позволяет управлять порядком их обработки.

Группировка приоритетов позволяет более гибко управлять обработкой прерываний в сложных системах.

Приложение 1

Листинг 1: Код программы

```
#include "stm32f2xx.h"                    // Device header
#include "core_cm3.h"
void delay ()
{
unsigned long i;
i=0;
for(i=0; i<2000000; i++){}
}

void EXTI0_IRQHandler(void)
{
GPIOG->ODR |= 1ul<<6;
delay ();
GPIOG->ODR &= ~1ul<<6;
delay ();

        EXTI->PR|=EXTI_PR_PR0;
}

void EXTI15_10_IRQHandler(void)
{
GPIOG->ODR |= 1ul<<8;
delay ();
GPIOG->ODR &= ~1ul<<8;
delay ();

        EXTI->PR|=EXTI_PR_PR0;
}


int main ()
{ RCC->AHB1ENR |= 1ul<<6; // Enable port G clocking
   RCC->APB2ENR|= 1ul<<14; //SYSCFGEN

  GPIOG->MODER = (GPIOG->MODER & ~(1ul<<13)) | 1ul<<12;//PG6
  GPIOG->MODER = (GPIOG->MODER & ~(1ul<<15)) | 1ul<<14;//PG7
  GPIOG->MODER = (GPIOG->MODER & ~(1ul<<17)) | 1ul<<16;//PG8

  GPIOG->MODER = (GPIOG->MODER & ~(1ul<<31)) & ~(1ul<<30);//PG15
  GPIOA->MODER = (GPIOA->MODER & ~(1ul<<1)) & ~(1ul);//PA0

  EXTI->IMR|=EXTI_IMR_MR0|EXTI_IMR_MR15;
//                                                                                (

  EXTI->RTSR|= EXTI_RTSR_TR0; //Rise Signal
  EXTI->FTSR|=EXTI_FTSR_TR15; //Fall Signal

  SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PA;
//                       pa0

SYSCFG->EXTICR[3] |= SYSCFG_EXTICR4_EXTI15_PG;

NVIC_SetPriority(6,5); //(
NVIC_SetPriority(40,6);
```

```
NVIC_EnableIRQ(6); //
NVIC_EnableIRQ(40);

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