МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий

Высшая школа искусственного интеллекта

Отчёт по лабораторной работе №3 по дисциплине

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ

на тему:

События и прерывания

Работу выполнил:

студент группы 5130201/20001

Тугай В.В.

Преподаватель:

Вербова Н. М.

Санкт-Петербург

2024

# Содержание

1. Цель
2. Постановка задачи
3. Алгоритм программы
4. Полученные результаты
5. Вывод
6. Приложение

**Цель**

Ознакомление с основными методами обработки событий и прерываний.

**Постановка задачи**

Используя библиотеки Keil μVision5, разработать программу для микроконтроллера (МК) STM32F200, которая включает и выключает светодиод.

**Алгоритм программы**

Для начала в данной лабораторной работе необходимо настроить выводы PG6,PG7 и PG8 на вывод цифровых данных аналогично предыдущим лабораторным работам: подключить тактирование, с целью снижения энергопотребления МК после подачи на него питания, так как до этого момента практически все периферийные блоки выключены. Включение/выключение блока производится подачей/прекращением на него тактового сигнала.

Управление режимами работы разрядов портов ввода/вывода общего назначения выполняется с помощью регистра режимов (См. Рисунок 1). В данном случае нам следует установить значение 01 для вывода цифровых данных.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 1

Выводы PA0 и PG15 настраиваются на ввод цифровых данных. Для этого также воспользуемся таблицей на рис. 2. Исходя из этого устанавливается значения 00, после подключения тактирования недостающих регистров.

После чего подключается тактирование контроллера конфигурации системы SYSCFG. Для ее подключения необходимо использовать конструкцию “Bit definition for RCC\_APB2ENR register” файла stm32f207xx.h. А для установки значения воспользоваться документацией из файла CD00225773.pdf, в файле была найдена информация о том, что для подключения тактирования необходимо перевести бит по номером 14 в состояние со значением 1, что и было выполнено непосредственно в самой программе.

Далее настраивается контроллер внешних событий/прерываний EXTI(External interrupt/event controller). Всего есть 23 линии для конфигурирования источников прерываний, однако мы будем использовать только линии 0 и 15.

Конфигурируются маскирующие биты для требуемых линий, то есть 0 и 15. Для этого обратимся к файлу stm32f207xx.h и использовать конструкции из “Bit definition for EXTI\_IMR register”.

На данном этапе произошла резервация двух линий под прерывания, далее необходимо сконфигурировать биты регистров выбора фиксации линий прерывания по нарастающему и падающему фронтам (EXTI\_RTSR и EXTI\_FTSR). Для этого обратимся к конструкциям “Bit definition for EXTI\_RTSR register” и “Bit definition for EXTI\_FTSR register”. В данной лабораторной работе происходит фиксация в одном случае нарастающего фронта(PA0), а в другом - падающего(PG15).

После этого прокрепляются порты к зарезервированным линиям. Это делается при помощи находящихся в контроллере конфигурации системы SYSCFG регистров конфигурации внешних прерываний контроллера конфигурации системы SYSCFG\_EXTICR. Для подключения следует воспользоваться конструкцией из “EXTI0 configuration” и “EXTI15 configuration” файла stm32f207xx.h.

После настройки контроллера внешних прерываний следует установить приоритеты прерываний. Установка приоритетов осуществляется с помощью регистров приоритета прерывания NVIC\_IPR (см. файл CD00228163.pdf). Для каждой 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 наконец активируется 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 каждой линии использовалась функция “void NVIC\_EnableIRQ(IRQn\_Type IRQn)” из строки 1382 файла core\_cm3.h.

Сам обработчик прерываний написан отдельной переопределенной функцией. Имя этой функции найдено в предустановленном файле StartUp. Для каждой линии обработчик прерываний индивидуален. Он включает и выключает прикрепленный к нему светодиод.

**Полученные результаты**

Реализованная на языке C программа (см. Приложение) успешно выполняет поставленную задачу.

**Вывод**

Были изучены основные методы обработки событий и прерываний. Разработана программа 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 подключается светодиод PG6, а при нажатии кнопки “USER” светодиод PG8.

**Приложение**

#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;

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

EXTI->RTSR|= EXTI\_RTSR\_TR0; //Rise Signal

EXTI->FTSR|=EXTI\_FTSR\_TR15; //Fall Signal

​

SYSCFG->EXTICR[0] |= SYSCFG\_EXTICR1\_EXTI0\_PA;

//прикрепили pa0 к зарезарвированной линии

​

SYSCFG->EXTICR[3] |= SYSCFG\_EXTICR4\_EXTI15\_PG;

​

NVIC\_SetPriority(6,5); //(номер в таблице векторов прерываний(тип), приоритет)

NVIC\_SetPriority(40,6);

​

NVIC\_EnableIRQ(6); //активировали прерывание

NVIC\_EnableIRQ(40);

​

for (;;)

{

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

delay ();

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

delay ();

}

}