Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о практической работе №5

«Обработка прерываний в контроллерах 1986ВЕ9x. Внешние прерывания EXT\_INT. Системный таймер»

по дисциплине

«Специализированные ЭВМ»

Выполнили:

Студенты группы 045

Вашкулатов Н.А.

Анохин В.А.

Проверил:

доц. каф. ЭВМ Устюков Д.И.

**Цель работы**: получить навыки работы с контроллером обработки прерываний NVIC.

**Ход работы**

**Задание 5.1.** Создайте проект по алгоритму из работы 1. Опишите процедуру обработчика указанного прерывания.

В тело подпрограммы добавьте код изменения состояния светодиода 1 через полсекунды на противоположное. Для упрощения реализации задержки в полсекунды можно дополнительно описать функцию программной задержки (аналогично примеру из задания 1.1).

В основной программе обеспечьте тактирование используемых портов, инициализацию портов, глобальное разрешение прерываний, разрешение прерывания EXT\_INT1.

В бесконечном цикле программы добавьте код изменения состояния светодиода 0 через полсекунды на противоположное.

**Код программы:**

#include "MDR32F9Qx\_port.h"// подключение заголовочного файла библиотеки

#include "MDR32F9Qx\_rst\_clk.h" // подключение заголовочного файла библиотеки

#include "MDR32Fx.h"

static PORT\_InitTypeDef LedInit;

static PORT\_InitTypeDef InputInit;

const int delay = 8000000;

const uint8\_t LED\_ZERO = PORT\_Pin\_0;

const uint8\_t LED\_ONE = PORT\_Pin\_1;

void Delay(int del) //процедура задржки del - кол-во циклов

{

for (int i=0; i<del; i++); //1 цикл – n тактов процессора

}

void LedConfig (void) //Процедура инициализации

{

LedInit.PORT\_Pin = (PORT\_Pin\_0 | PORT\_Pin\_1); //вывод 0

LedInit.PORT\_FUNC = PORT\_FUNC\_PORT; //режим порт

LedInit.PORT\_MODE = PORT\_MODE\_DIGITAL; //цифровой

LedInit.PORT\_OE = PORT\_OE\_OUT; //сигнал на вывод

LedInit.PORT\_SPEED = PORT\_SPEED\_SLOW; //медленный фронт

PORT\_Init(MDR\_PORTC, &LedInit); //инициализация параметров

}

void InputConfig(){

InputInit.PORT\_Pin = PORT\_Pin\_0;

InputInit.PORT\_FUNC = PORT\_FUNC\_ALTER;

InputInit.PORT\_OE = PORT\_OE\_IN;

InputInit.PORT\_SPEED = PORT\_SPEED\_FAST;

InputInit.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;

InputInit.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_ON;

PORT\_Init(MDR\_PORTA, &InputInit);

}

void CPU\_Config(){

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

//запрос состояния, только если готов, то переход к следующим действиям

while (RST\_CLK\_HSEstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul6);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void changeLedStatus(uint8\_t led){

if(PORT\_ReadInputDataBit(MDR\_PORTC, led)){

PORT\_ResetBits(MDR\_PORTC, led);

} else {

PORT\_SetBits(MDR\_PORTC, led);

}

}

void IntConfig(){

NVIC\_EnableIRQ(EXT\_INT1\_IRQn);

\_\_enable\_irq();

}

void EXT\_INT1\_IRQHandler(void){

Delay(delay/5);

changeLedStatus(LED\_ZERO);

Delay(delay/5);

changeLedStatus(LED\_ZERO);

Delay(delay/5);

changeLedStatus(LED\_ZERO);

Delay(delay/5);

changeLedStatus(LED\_ZERO);

Delay(delay);

}

int main(){

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE);//разрешение тактирования PORTC

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);//разрешение тактирования PORTA

CPU\_Config();

LedConfig();

InputConfig();

IntConfig();

uint32\_t count = 0;

while(1){

if(count == delay){

changeLedStatus(LED\_ONE);

count = 0;

}

count++;

}

}

Во время работы программы мигает светодиод 0. При нажатии на кнопку возникает прерывание и начинает мигать светодиод 1. Светодиод 0 остается в том же состоянии, что и на момент нажатия кнопки. После того как отпустили кнопку светодиод 1 гаснет, а светодиод 0 возобновляет мигание.

**Задание 5.2.** В программе из задания 1 сделайте следующие изменения:

Переопределите выводы PC0 (PF0) на PA1, PC1 (PF1) на PA2 в процедуре инициализации, обработчике прерываний, в главной программе, уберите связанные с ними программные задержки, также проинициализируйте на вывод PA3.

Добавьте в подпрограмму main (до бесконечного цикла) процедуру инициализации системного таймера SysTick\_Config(uint32\_t ticks).

Обеспечьте разрешение прерывания SysTick (см. таблицы 5.3 и 5.4).

Опишите процедуру обработчика прерывания SysTick. В теле процедуры обработчика опишите изменение значения PA3 на противоположное.

Подключите логический анализатор к выводам PA1, PA2, PA3.

Проанализируйте обработку прерываний в различных ситуациях с помощью логического анализатора.

**Код программы:**

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

static PORT\_InitTypeDef LedInit;

static PORT\_InitTypeDef InputInit;

const int delay = 10000000;

const uint8\_t LED\_ZERO = PORT\_Pin\_1;

const uint8\_t LED\_ONE = PORT\_Pin\_2;

const uint8\_t LED\_TWO = PORT\_Pin\_3;

void Delay(int del)

{

for (int i=0; i<del; i++);

}

void OutputConfig (void)

{

LedInit.PORT\_Pin = (PORT\_Pin\_1 | PORT\_Pin\_2 | PORT\_Pin\_3);

LedInit.PORT\_FUNC = PORT\_FUNC\_PORT;

LedInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

LedInit.PORT\_OE = PORT\_OE\_OUT;

LedInit.PORT\_SPEED = PORT\_SPEED\_FAST;

PORT\_Init(MDR\_PORTA, &LedInit);

}

void InputConfig(){

InputInit.PORT\_Pin = PORT\_Pin\_0;

InputInit.PORT\_FUNC = PORT\_FUNC\_ALTER;

InputInit.PORT\_OE = PORT\_OE\_IN;

InputInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

InputInit.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_ON;

PORT\_Init(MDR\_PORTA, &InputInit);

}

void CPU\_Config(){

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul6);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void changeLedStatus(uint8\_t led){

if(PORT\_ReadInputDataBit(MDR\_PORTA, led)){

PORT\_ResetBits(MDR\_PORTA, led);

} else {

PORT\_SetBits(MDR\_PORTA, led);

}

}

void IntConfig(){

NVIC\_EnableIRQ(EXT\_INT1\_IRQn);

\_\_enable\_irq();

// NVIC\_EnableIRQ(SysTick\_IRQn);

//SysTick\_Config(delay);

}

void SysTick\_Handler(void){

changeLedStatus(LED\_TWO);

}

void EXT\_INT1\_IRQHandler(void){

Delay(delay);

changeLedStatus(LED\_ONE);

}

int main(){

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);

CPU\_Config();

OutputConfig();

InputConfig();

IntConfig();

uint32\_t count = 0;

while(1){

Delay(delay);

changeLedStatus(LED\_ZERO);

}

}

Во время работы программы начинает постоянно возникать прерывание INT. Возникающие прерывания INT имеют приоритет над SysTick и поэтому постоянно мигает светодиод 1.

**Задание 5.3.** В программе, полученной в задании 2, задайте приоритеты прерываниям EXT1 и SysTick, используя функцию void NVIC\_SetPriority (IRQn\_t IRQn, uint32\_t priority). Проанализируйте работу системы с двумя работающими прерываниями без задания приоритетов, с большим приоритетом для SysTick, а затем c большим приоритетом для EXT1. Выполните сборки проекта. Проанализируйте обработку прерываний при различном порядке возникновения прерываний с помощью логического анализатора.

**Код программы:**

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

static PORT\_InitTypeDef LedInit;

static PORT\_InitTypeDef InputInit;

const int delay = 4000000;

const uint8\_t LED\_ZERO = PORT\_Pin\_1;

const uint8\_t LED\_ONE = PORT\_Pin\_2;

const uint8\_t LED\_TWO = PORT\_Pin\_3;

void Delay(int del)

{

for (int i=0; i<del; i++);

}

void OutputConfig (void)

{

LedInit.PORT\_Pin = (PORT\_Pin\_1 | PORT\_Pin\_2 | PORT\_Pin\_3);

LedInit.PORT\_FUNC = PORT\_FUNC\_PORT;

LedInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

LedInit.PORT\_OE = PORT\_OE\_OUT;

LedInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init(MDR\_PORTA, &LedInit);

}

void InputConfig(){

InputInit.PORT\_Pin = PORT\_Pin\_0;

InputInit.PORT\_FUNC = PORT\_FUNC\_ALTER;

InputInit.PORT\_OE = PORT\_OE\_IN;

InputInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

InputInit.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_ON;

InputInit.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;

InputInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_Init(MDR\_PORTA, &InputInit);

}

void CPU\_Config(){

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul6);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

while (RST\_CLK\_CPU\_PLLstatus()!=SUCCESS);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

void changeLedStatus(uint8\_t led){

if(PORT\_ReadInputDataBit(MDR\_PORTA, led)){

PORT\_ResetBits(MDR\_PORTA, led);

} else {

PORT\_SetBits(MDR\_PORTA, led);

}

}

void IntConfig(){

\_\_enable\_irq();

// NVIC\_EnableIRQ(EXT\_INT1\_IRQn);

NVIC\_EnableIRQ(SysTick\_IRQn);

SysTick\_Config(delay);

//Same

//NVIC\_SetPriority(EXT\_INT1\_IRQn, 1);

//NVIC\_SetPriority(SysTick\_IRQn, 1);

//Timer

NVIC\_SetPriority(EXT\_INT1\_IRQn, 2);

NVIC\_SetPriority(SysTick\_IRQn, 1);

//Btn

//NVIC\_SetPriority(EXT\_INT1\_IRQn, 1);

//NVIC\_SetPriority(SysTick\_IRQn, 2);

}

void SysTick\_Handler(void){

changeLedStatus(LED\_TWO);

}

void EXT\_INT1\_IRQHandler(void){

Delay(delay);

changeLedStatus(LED\_ONE);

}

int main(){

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);

CPU\_Config();

OutputConfig();

InputConfig();

IntConfig();

uint32\_t count = 0;

while(1){

if(count == delay){

changeLedStatus(LED\_ZERO);

count = 0;

}

count++;

}

}

При задании повышенного приоритета для SysTick начинают мигать оба светодиода.

**Задание 5.4.** В главном цикле программы из задания 1 замените фрагмент бесконечного цикла в подпрограмме main на фрагмент, представленный ниже.

|  |
| --- |
| while(1) //основной цикл программы  {  for (int i=0; i<5; i++){  PORT\_SetBits(MDR\_PORTC, PORT\_Pin\_0);  delay (500000);  PORT\_ResetBits(MDR\_PORTC, PORT\_Pin\_0);  delay (500000);  }  \_\_WFI();//режим ожидания до выполнения прерывания  } |

Проанализируйте обработку прерываний.

Светодиод загорается 5 раз. После этого происходит ожидание нажатия кнопки. После нажатия светодиод мигает 5 раз.

**Вывод**: в ходе выполнения работы были получены навыки работы с контроллером обработки прерываний NVIC.