МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

кафедра «Вычислительная техника»

**Лабораторная работа №3 (вариант 5)**

**Тема: «Реализация прерываний микроконтроллера»**

по дисциплине: «Микропроцессорные системы»

Выполнил:

студент 4 курса, гр. ИВТВМбд-41

Захарычев Н.А

Проверил:

к.т.н, доцент кафедры ВТ

Игонин А.Г.

г. Ульяновск, 2017

**Содержание**

Содержание………………………………………………………………………..2

Задание …………………………..…………………………………………….….3

Ход работы………………………………………………………………………..3

Тестирование……………………………………………………………………...6

Вывод……………………………………………………………………………...7

Список использованной литературы……………………………………………7

Приложение. Исходный код……………………………………………………..8

**Задание**

Необходимо реализовать работу лабораторной работы №2 с применением прерываний микроконтроллера.

Функции, заданные по варианту:

O1 = (!02 & !03 & !04) & T3

O2 = (I4 | O4 | I3) & T1

O3 = (I1 & I2) & T2

O4 = O3 & I4

Характеристики таймеров:

T1 = 5, T2 = 7, T3 = 9

Микроконтроллер по варианту:

Семейство микроконтроллеров AVR. Atmega8, 8MHz

**Ход работы**

Одним из преимуществ микроконтроллера ATmega8 является широкий диапазон различных прерываний.

Прерывание представляет собой событие, при наступлении которого выполнение основной программы приостанавливается и вызывается функция, обрабатывающая прерывание определённого типа.

Прерывания делятся на внутренние и внешние. К источникам внутренних прерываний относятся встроенные модули микроконтроллера (таймеры, приёмопередатчик USART и т.д). Внешние прерывания возникают при поступлении внешних сигналов на выводы микроконтроллера (например сигналы на выводы RESET и INT).

За управление прерываниями в ATmega8 отвечают 4 регистра:

* GIMSK (он же GICR) - запрет/разрешение прерываний по сигналам на входах INT0, INT1
* GIFR - управление всеми внешними прерываниями
* TIMSK, TIFR - управление прерываниями от таймеров/счётчиков

В ходе работы были выбраны следующие прерывания, представленные в таблице №1.

**Таблица №1**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Адрес** | **Источник прерывания** | **Описание** |
| 1 | 0x0002 | INT1 | Внешний запрос на прерывание по входу INT1 |
| 2 | 0x0004 | T/C1 | Совпадение с регистром сравнения A таймера T/C1 |
| 3 | 0x0009 | UART, RXC | Приём данных приёмопередптчиком UART завершен |

**Входные и выходные сигналы**

Для входных сигналов был выбран порт B. В качестве высокого уровня сигнала на входе используется питание микроконтроллера. В данном случае биты отвечают за потдягивание к соответствующим ножкам порта резисторов на шину питания.

// Устаналиваем PORTB на режим чтения

DDRB = 0x00;

PORTB = 0x00;

Считывание входных сигналов происходит с регистра PINB:

int inp = PINB;

i1 = inp & (1<<PB0);

i2 = inp & (1<<PB1);

i3 = inp & (1<<PB2);

i4 = inp & (1<<PB3);

Для выходных сигналов выбран порт C. В качестве самих сигналов выступают светодиоды.

// Устаналиваем PORTС на режим записи

DDRC = 0xFF;

PORTC = 0x00;

Реализация получения выходных сигналов:

void setLeds(void) {

o1 = (!o2 && !o3 && !o4) && t3;

o2 = (i4 || o4 || i3) && t1;

o3 = (i1 && i2) && t2;

o4 = o3 && i3;

PORTC = o1 ? PORTC | (1<<PC0) : PORTC & ~(1<<PC0);

PORTC = o2 ? PORTC | (1<<PC1) : PORTC & ~(1<<PC1);

PORTC = o3 ? PORTC | (1<<PC2) : PORTC & ~(1<<PC2);

PORTC = o4 ? PORTC | (1<<PC3) : PORTC & ~(1<<PC3);

}

**Обработка прерываний таймера и INT1**

TIMER1\_COMPA\_vect - прерывание совпадения TIMER1 с числом A. Когда таймер достигает заданное время – происходит прерывание, в результате которого загораются светодиоды (выходные сигналы). Реализация прерывания приведена ниже:

ISR (TIMER1\_COMPA\_vect)

{

i++;

if (i % 5 == 0) t1 = !t1;

if (i % 7 == 0) t2 = !t2;

if (i % 9 == 0) t3 = !t3;

setLeds();

}

INT1\_vect - внешнее прерывание, реализовано через нажатие на кнопку. При нажатии на кнопку происходит прерывание, которое сохраняет текущие выходные сигналы, далее все выходные сигналы (светодиоды) моргают 5 секунд, затем возвращаем сохраненное до прерывания состояние. Реализация прерывания приведена ниже:

ISR(INT1\_vect)

{

unsigned char i, temp;

\_delay\_ms(500); // Задержка

temp = DataPort; // Сохраняем текущее значение лампочек

// Мигаем лампочками 5секунд

for(i = 0; i<5; i++)

{

DataPort = 0x00;

\_delay\_ms(500); // Задержка

DataPort = 0xFF;

\_delay\_ms(500); // Задержка

}

DataPort = temp; // Вовзаращаем исходное значение лампочек

}

**Обмен данными по uart c использованием прерываний RXC**

ISR(USART\_RXC\_vect) – прерывание по окончанию принятия байта. Данное прерывание происходит когда пользователь заканчивает вводить какие-либо символы с клавиатуры (терминала), тогда текущие выходные сигналы меняются на вновь пришедшие от прерывания, а затем восстанавливаются. Для начала необходимо инициализировать UART:

void USART\_Init(void){

// Устанавливаем скорость передачи

UBRRL = BAUD\_PRESCALE;

UBRRH = (BAUD\_PRESCALE >> 8);

// Разрешаем получение и передачу

UCSRB = ((1<<TXEN)|(1<<RXEN) | (1<<RXCIE));

}

Реализация передачи байта:

void USART\_SendByte(*uint8\_t* u8Data){

// Ждем пока последний байт будет передан

while((UCSRA &(1<<UDRE)) == 0);

// Отправляем данные

UDR = u8Data;

}

Реализация самого прерывания:

ISR(USART\_RXC\_vect){

value = UDR; // чтение UART`om значения

DataPort = 0xFF; // включаем все выходные лампы

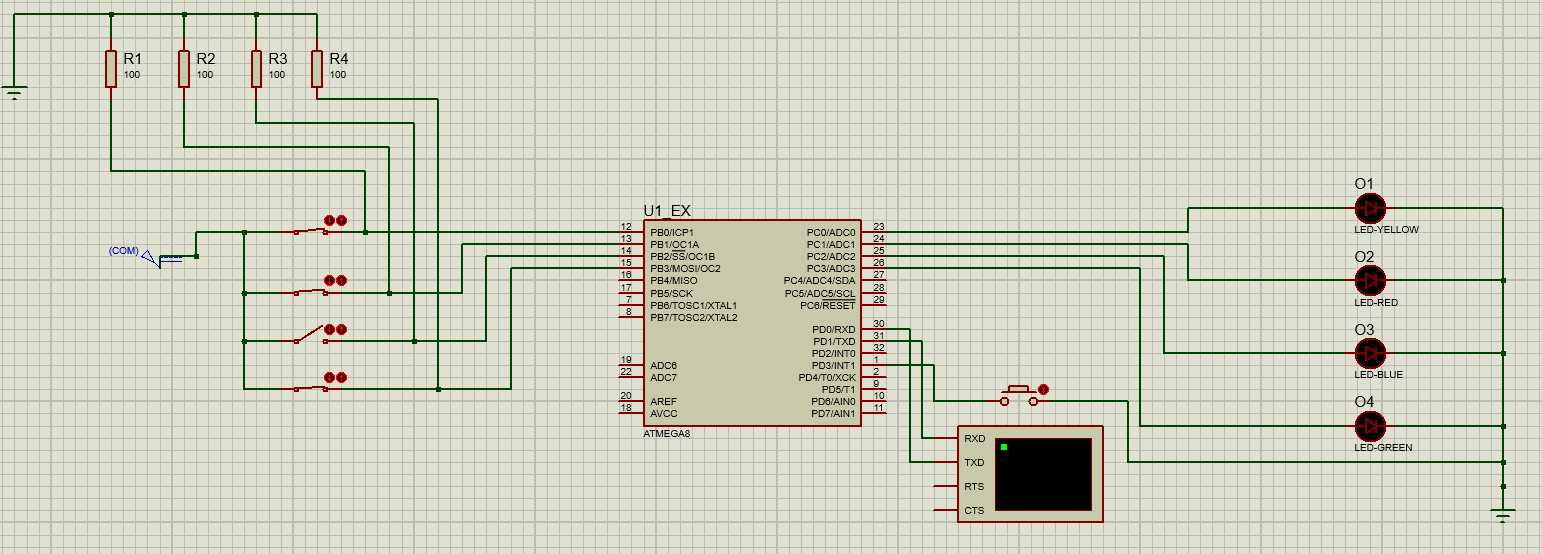
//USART\_SendByte('1');

}

Считываем значение с отправленного байта, если конец, то вызываем прерывание, которое включает все выходные сигналы.

Командой sei() необходимо разрешить глобальные прерывания.

**Общая схема в Proteus8**



**Рисунок 1. Общая схема**

**Тестирование**

В результате тестирования было выявлено, что программа работает корректно, а именно: процессор незамедлительно выполняет прерывания, а затем возвращается в свое исходное состояние. Также процессор запоминает состояния, чтобы избежать аварийных состояний.

**Выводы**

В ходе выполнения данной лабораторной работы были изучены вектора прерываний Atmega8, получены навыки настройки микроконтроллера, а также реализации и обработки прерываний.

**Список литературы**

1. Описание прерываний микроконтроллера ATmega8. [Электронный ресурс] http://www.atmega8.ru/wiki/view/doc.9.html (дата обращения: 08.12.2017)
2. Инициализация и работа с UART. [Электронный ресурс] http://www.customelectronics.ru/avr-rabota-s-uart/ (дата обращения: 08.12.2017)
3. Работа с прерываниями INT0 и INT1. [Электронный ресурс] https://avrlab.com/node/37 (дата обращения: 08.12.2017)

**Приложение. Исходный код**

#define F\_CPU 8000000

#define DataPort PORTC // Using PortC as our Dataport

#define DataDDR DDRC

#define USART\_BAUDRATE 38400

#define BAUD\_PRESCALE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)

#include <avr/io.h>

#include <util/delay.h>

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include <time.h>

#include <avr/interrupt.h>

int i;

// Создаем I, заданные по варианту

int i1 = 0, i2 = 0, i3 = 0, i4 = 0;

// создаем таймеры, заданные по вараинту

int t1 = 0, t2 = 0, t3 = 0;

// значения функций, создание лампочек

int o1 = 0, o2 = 0, o3 = 0, o4 = 0;

volatile unsigned char value;

void initTimer() {

MCUCR |= (1 << ISC10);

GICR |= (1 << INT1); // Включаем прерывание по INT1

TCCR1B |= (1<<WGM12);

TIMSK |= (1<<OCIE1A);

OCR1AH = 0b01111010;

OCR1AL = 0b00010010;

TCCR1B |= (1<<CS12);

}

void USART\_Init(void){

// Устанавливаем скорость передачи

UBRRL = BAUD\_PRESCALE;

UBRRH = (BAUD\_PRESCALE >> 8);

// Разрешаем получение и передачу

UCSRB = ((1<<TXEN)|(1<<RXEN) | (1<<RXCIE));

}

void setLeds(void) {

o1 = (!o2 && !o3 && !o4) && t3;

o2 = (i4 || o4 || i3) && t1;

o3 = (i1 && i2) && t2;

o4 = o3 && i3;

PORTC = o1 ? PORTC | (1<<PC0) : PORTC & ~(1<<PC0);

PORTC = o2 ? PORTC | (1<<PC1) : PORTC & ~(1<<PC1);

PORTC = o3 ? PORTC | (1<<PC2) : PORTC & ~(1<<PC2);

PORTC = o4 ? PORTC | (1<<PC3) : PORTC & ~(1<<PC3);

}

void USART\_SendByte(uint8\_t u8Data){

// Ждем пока последний байт будет передан

while((UCSRA &(1<<UDRE)) == 0);

// Отправляем данные

UDR = u8Data;

}

ISR(USART\_RXC\_vect){

value = UDR; // чтение UART`om значения

DataPort = 0xFF; // включаем все выходные лампы

//USART\_SendByte('1');

}

ISR (TIMER1\_COMPA\_vect)

{

i++;

if (i % 5 == 0) t1 = !t1;

if (i % 7 == 0) t2 = !t2;

if (i % 9 == 0) t3 = !t3;

setLeds();

}

ISR(INT1\_vect)

{

unsigned char i, temp;

\_delay\_ms(500); // Задержка

temp = DataPort; // Сохраняем текущее значение лампочек

// Мигаем лампочками 5секунд

for(i = 0; i<5; i++)

{

DataPort = 0x00;

\_delay\_ms(500); // Задержка

DataPort = 0xFF;

\_delay\_ms(500); // Задержка

}

DataPort = temp; // Вовзаращаем исходное значение лампочек

}

uint8\_t USART\_ReceiveByte(){

while((UCSRA &(1<<RXC)) == 0);

return UDR;

}

int main (void)

{

initTimer();

USART\_Init();

// Устаналиваем PORTB на режим чтения

DDRB = 0x00;

PORTB = 0x00;

// Устаналиваем PORTС на режим записи

DDRC = 0xFF;

PORTC = 0x00;

MCUCR |= (1 << ISC10); // устанавливаем INT1 на работу по срезу

i = 0;

sei();

i1 = PINB&(1<<0);

i2 = PINB&(1<<1);

i3 = PINB&(1<<2);

i4 = PINB&(1<<3);

setLeds();

//PORTC = ~value; // 0 = LED on

while(1) {

int inp = PINB;

i1 = inp & (1<<PB0);

i2 = inp & (1<<PB1);

i3 = inp & (1<<PB2);

i4 = inp & (1<<PB3);

//USART\_SendByte(value); // send value

\_delay\_ms(250);}