Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Автоматизация проектирования микропроцессорных систем»

Лабораторная работа №4

Вариант № 17

Выполнил студент

группы ИВТАСбд-42:

Сулейманов М.З.

Проверил:

Игонин А.Г.

Ульяновск, 2024

**Оглавление**

[1. Задание 3](#_Toc186078151)

[2. Краткие теоретические сведения 3](#_Toc186078152)

[3. Основные сведения об устройстве 3](#_Toc186078153)

[4. Сведения о среде моделирования 4](#_Toc186078154)

[5. Порядок выполнения работы 5](#_Toc186078155)

[6. Вывод 10](#_Toc186078156)

[7. Список литературы 10](#_Toc186078157)

[8. Приложение 11](#_Toc186078158)

# Задание

Разобраться с поддерживаемыми прерываниями. Реализовать три обработчика прерывания: USART Data Register Empty, EEPROM Ready, External Interrupt Request 2 (INT2).

# Краткие теоретические сведения

Микропроцессорные системы — частный вид электронный системы, которые обладают узлом, блоком, прибором ил комплексом, которые производят обработку информации.

Сигнал — это изменение во времени некоторой физической величины.

Микроконтроллер — микросхема для программного управления электронными устройствами.

Прерывания — это изменения в потоке управления, вызванные не самой программой, а чем-либо другим.

Обработчик прерываний — это специальная процедура, вызываемая по прерыванию для выполнения его обработки.

# Основные сведения об устройстве

Микроконтроллер ATmega16 - это 8-ми разрядный микроконтроллер с 16 Кбайт внутрисистемно программируемой Flesh-памяти. Схема и расположение выводов представлены на рис.1.

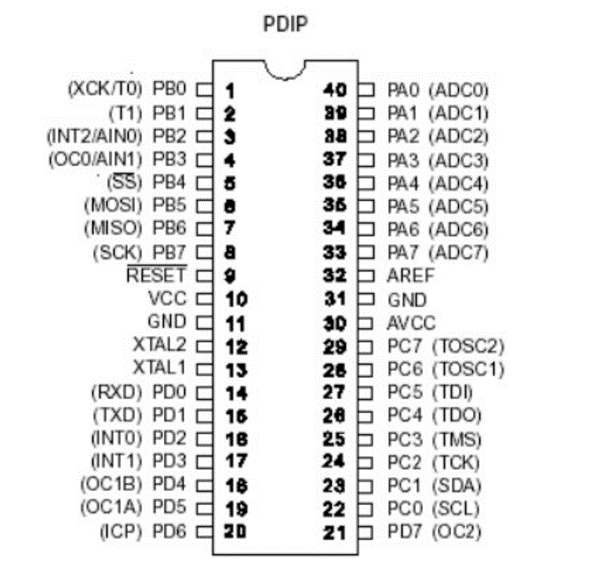


Рис. 1. Схема ATmega16

# Сведения о среде моделирования

Proteus – это пакет программ для автоматизированного проектирования (САПР) электронных схем.

Представляет возможности ввода схемы в графическом редакторе, моделирования ее работы и разработки печатной платы. Возможность моделирования работы микроконтроллеров (PIC, AVR и т. д.)

Microchip Studio - (ранее Atmel Studio и AVR Studio) — основанная на Visual Studio бесплатная проприетарная интегрированная среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel, работающая в операционных системах Windows. Atmel Studio содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер.

# Порядок выполнения работы

Для выполнения данной работы необходимо создать проект, в котором будет размещаться схема. После создания, необходимо добавить в список те элементы схемы, которые будут отображаться на схеме. В меню, во вкладке

«Component Mode» необходимо найти следующие элементы:

– LED-GREEN – светодиод для отображения результатов функций

– ATMEGA16 — микроконтроллер

– RES – резистор, который соединяется с выводом микроконтроллера и светодиодом. Для резистором надо поставить сопротивление 220 Ом.

Во вкладке «Terminals Mode» надо добавить элемент GROUND — для заземления светодиода.

По умолчанию питание микроконтроллера не отображается в Proteus, поэтому добавлять лишнее не имеет необходимости. В дальнейшем будут также описаны добавленные элементы, описанные выше — основные.

Обработчик прерываний USART Register Empty:

USART — от англ. Universal Synchronous/Asynchronous ReceiverTransmitter — Универсальный Синхронный/Асинхронный Приёмопередатчик.

Кроме линий передачи данных, может иметь отдельную линию для сигнала синхронизации. Главное отличие USART в том, что он может работать как в синхронном, так и в асинхронном режимах.

Когда USART получает байт данных, он сначала сохраняет его в буфере приемных данных. Затем, когда происходит следующий цикл передачи (например, после отправки предыдущего байта), USART проверяет, есть ли данные в буфере приемных данных для отправки. Если есть, они отправляются.

Когда все данные из буфера приемных данных отправлены, USART генерирует флаг RX Complete (RXC). Это означает, что все данные, которые были приложены к входному пину USART, были успешно обработаны и отправлены.

В проекте Proteus для осуществления данного обработчика прерывания был добавлен элемент схемы Virtual Terminal – это виртуальный терминал, который отображает ввод с клавиатуры. К пину с TX на терминале нужно подключить через провод порт микроконтроллера RXD(PD0) — с терминала считывается символ и заносит в буфер приемных данных микроконтроллера. К порту PB0 был подключен светодиод, который при каждом отправленном символе будет зажигаться и погасать.(Рис. 3)

При инициализации USART необходимо обозначить регистры(листинг 1):

1) UBBRH и UBBRL – установка частоты скорости передачи данных.

UBRRH содержит верхнюю часть значения, а UBRRL - нижнюю часть.

2) UCSRA - установка флага RX Complete (RXC). Когда все данные из буфера приемных данных отправлены, этот флаг устанавливается в 1.

3) UCSRB – режимы работы: RXCIE - разрешает генерацию прерывания

при завершении приема данных, RXEN - включает приемные данные.

4) UCSRC - формат данных: URSEL - указывает, что мы работаем с регистрами USART, UCSZ1 и UCSZ0 - определяют количество бит данных.

*Листинг 1. USART Register Empty*

|  |
| --- |
| void USART\_init(){  UBRRL = UBRRL\_value;  UBRRH = (UBRRL\_value >> 51);  UCSRB = (1 << RXEN) | (1 << TXEN);  UCSRC = (1<<UCSZ0)|(1<<URSEL);  }  void USART\_sendChar(char value)  {  while (!(UCSRA & (1 << UDRE)));  UDR = value;  UCSRB |= (1 << UDRIE);  }  ISR(USART\_UDRE\_vect)  {    PORTB = (1 << PB0);  \_delay\_ms(1000);  PORTB = (0 << PB0);  \_delay\_ms(1000);  UCSRB &= ~(1 << UDRIE);    } |

Обработчик прерываний EEPROM Ready:

EEPROM Ready - это цифровой блок на микроконтроллерах AVR, который генерирует выходной сигнал на основе данных энергонезависимой памяти, а именно: генерирует данные, когда регистр пуст и готов к приему данных.

Основные характеристики энергонезависимой памяти:

1) Она может работать как отдельный модуль или быть интегрирован в другие функции микроконтроллера.

2) Может использоваться для обнаружения изменений в аналоговых сигналах без необходимости использования АЦП.

3) Может генерировать прерывания для обработки событий сравнения.

Для реализации данного обработчика в проекте была подключена кнопка к порту PA1.

*Листинг 2. EEPROM Ready*

|  |
| --- |
| void EEPROM\_sendChar(char value) {  while (EECR & (1 << EEWE));  EECR |= (1 << EERIE);  eeprom\_write\_byte(0, value);  }  ISR(EE\_RDY\_vect) {  PORTB = (1 << PB1);  \_delay\_ms(1000);  PORTB = (0 << PB1);  \_delay\_ms(1000);  EECR &= ~(1 << EERIE);  } |

Обработчик прерываний External Interrupt Request 2 (INT2):

Данный обработчик был использован в лабораторной работе, но с режимом работы по совпадению. Режим работы по переполнению - таймер считает приходящие импульсы и при переполнении счётного регистра устанавливает флаг прерывания по переполнению. При этом счётный регистр сбрасывается в 0 и подсчет импульсов начинается сначала.

Единственный регистр, который поменялся: TIMSK |= (1 << TOIE2) - разрешает прерывание по событию переполнение.

Для данного обработчика на схему был добавлен светодиод к порту PD6, который зажигается раз в 3 секунды (листинг 3).

*Листинг 3. External Interrupt Request 2 (INT2)*

|  |
| --- |
| void INT2\_Init(void) {  MCUCSR &= ~(0 << ISC2);  GICR |= (1 << INT2);  }  ISR(INT2\_vect) {  PORTD = (1 << PD6);  } |

По итогу, была получена схема (Рис.3),

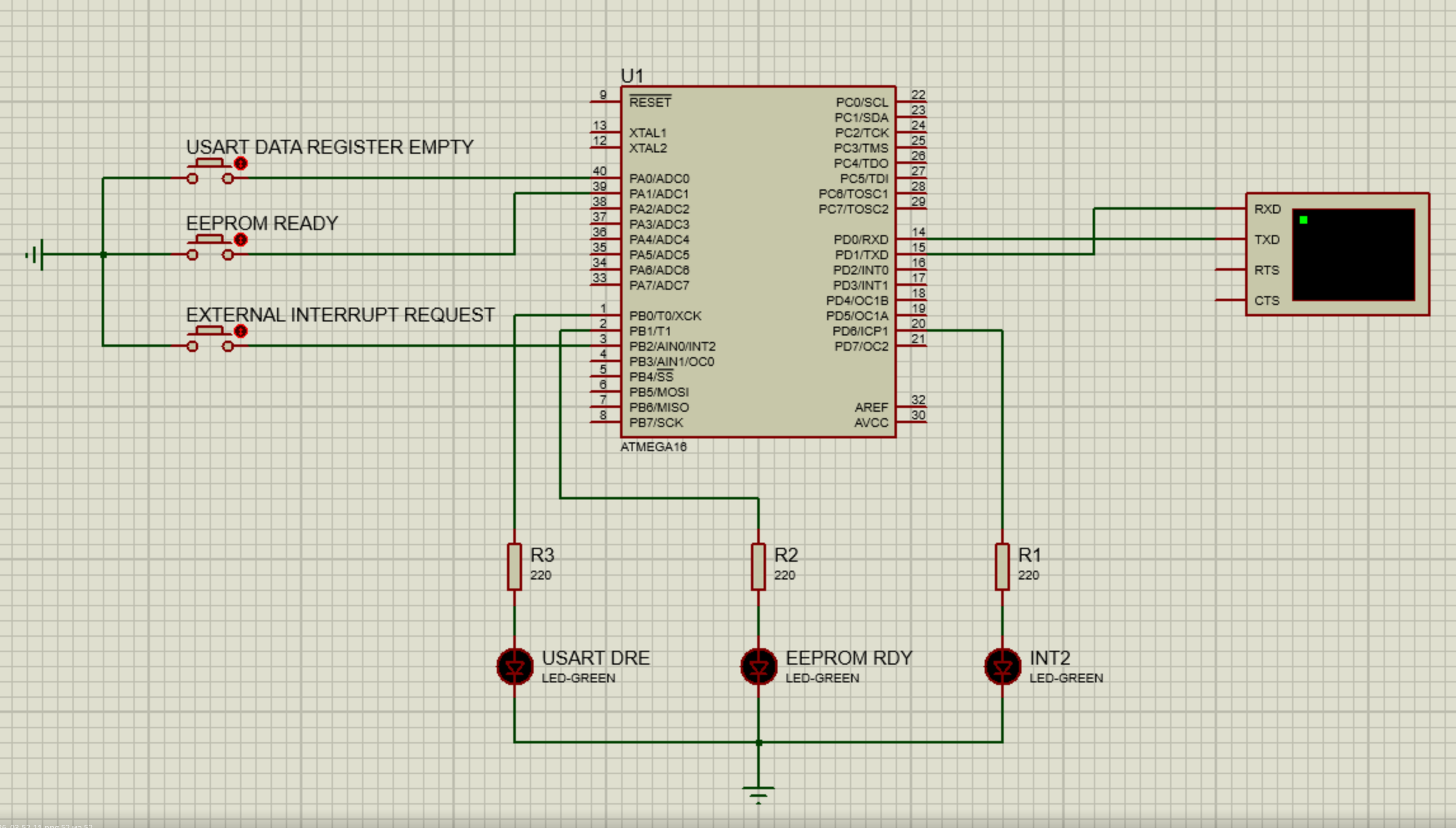
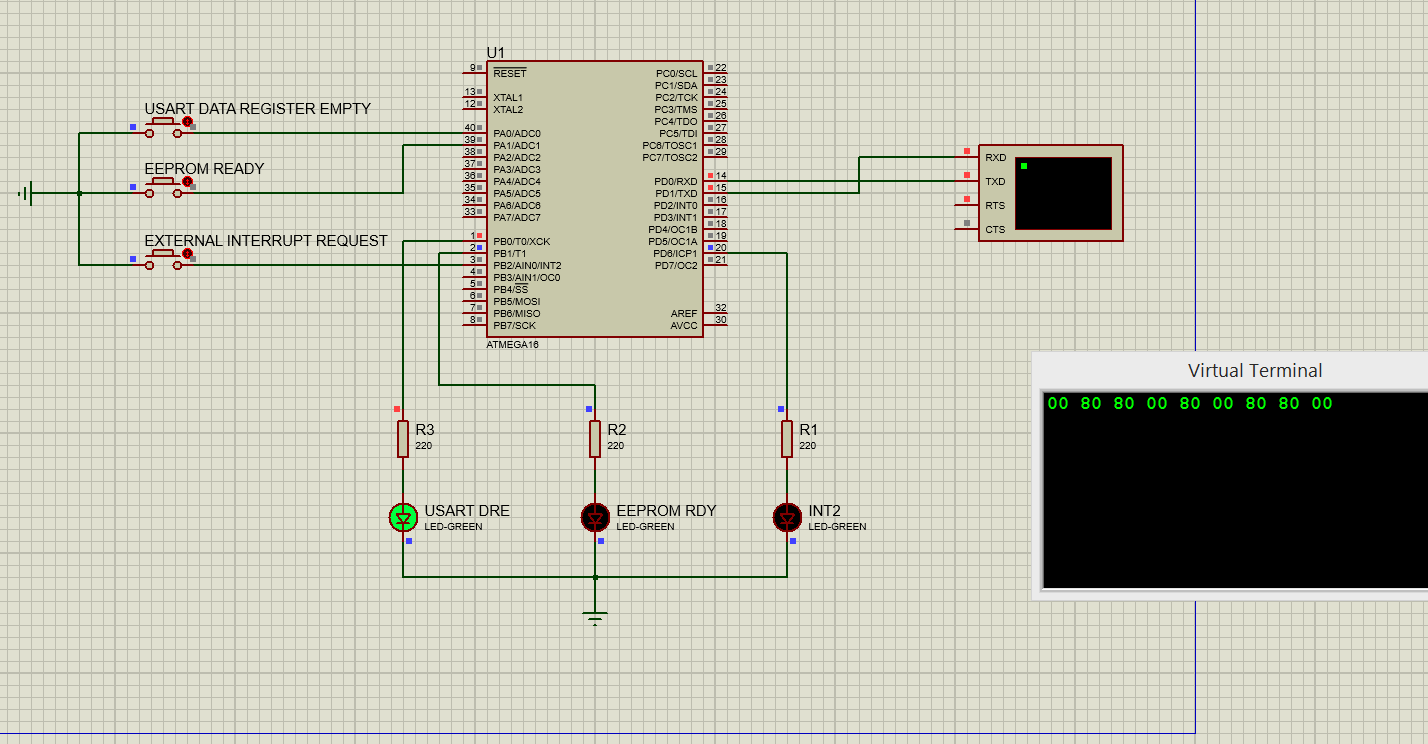


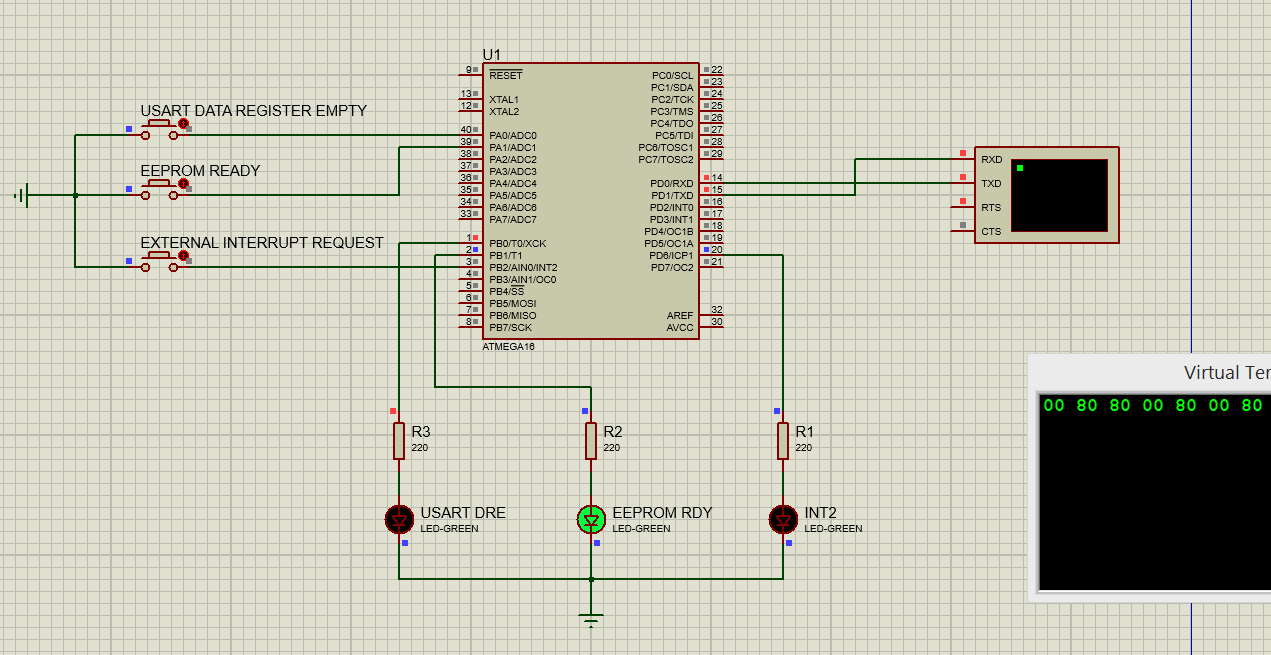
Рис. 3. Схема в Proteus

Тестирование:

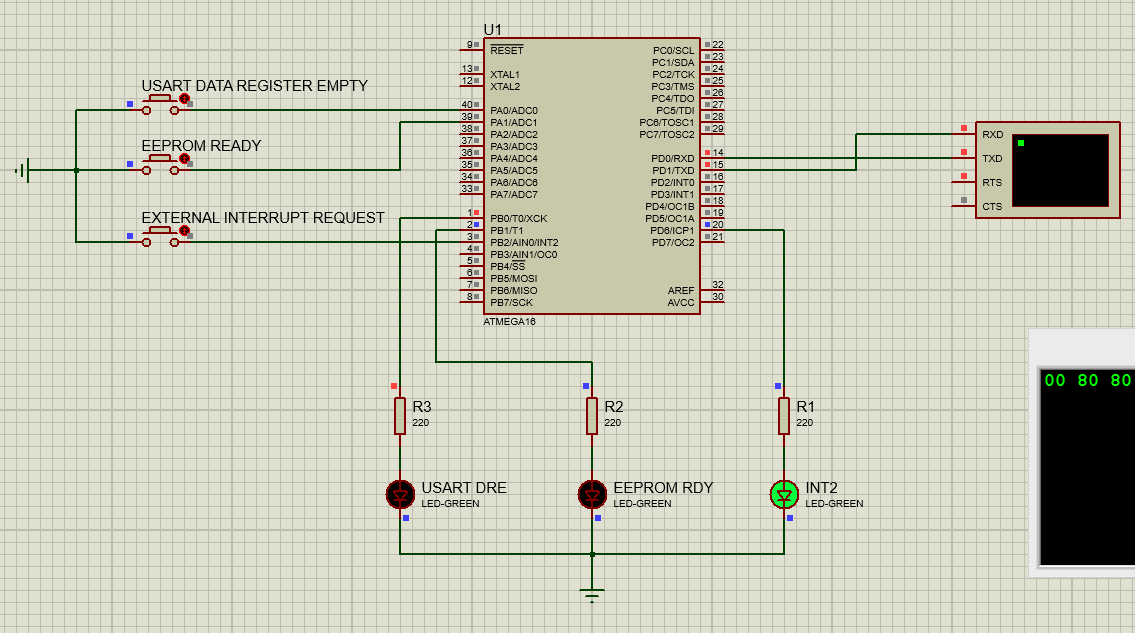
* USART Data Register Empty



* EEPROM Ready



* INT2



# Вывод

В ходе выполнения лабораторной работы, была разработана модель схемы, которая показывает работу обработчиков прерывания. Были изучены и реализованы три обработчика прерывания.

# Список литературы

1. Разработка и отладка микропроцессорных устройств в виртуальной среде

моделирования Proteus [Электронный ресурс]: метод. указания / сост. В. Г.

Иоффе. URL: http://repo.ssau.ru/bitstream/Metodicheskie-materialy/Razrabotka-iotladka-mikroprocessornyh-ustroistv-v-virtualnoi-srede-modelirovaniyaProteus-Elektronnyi-resurs-metod-ukazaniya70958/1/Иоффе%20В.Г.%20Разработка%20и%20отладка%20микропроцессорных%20устройств.pdf

2. Филатов М. Синтез цифровых устройств комбинационного типа в Proteus 8.1 | Компоненты и технологии [Электронный ресурс] - URL : https://kite.ru/proteus-8-1-2-2/

3. Базовые элементы в Proteus 8 | RXRX [Электронный ресурс] - URL : https://rxtx.su/proteus/bazovye-elementy-v-proteus-8/

4. Встроенный таймеры и счетчики AVR микроконтроллеров | VPAYAEM.RU [Электронный ресурс] URL : https://vpayaem.ru/Atmega8\_timer.html

5. Павел Бобков. Учебный курс AVR. Таймер - счетчик Т0. Регистры. Ч1 [Электронный ресурс] — URL: https://chipenable.ru/index.php/programming-avr/171-avr-timer-t0-ch1.html

6. ATmega16, ATmega16L [Электронный ресурс] - URL: http://www.gaw.ru/html.cgi/txt/ic/Atmel/micros/avr/atmega16.htm?ysclid=m2rc6fuz7m331298024

7. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е изд. — СПб.: Питер, 2013. — 816 с.: ил.

8. USART. Связь МК с ПК. [Электронный ресурс] - URL: <https://narodstream.ru/avr-urok-14-usart-svyaz-mk-s-pk-chast-5/>? ysclid=m2uoedh4xm932969594

9. AVR. Учебный курс. Использование аналогового компаратора [Электронный ресурс] — URL: https://easyelectronics.ru/avr-uchebnyj-kursispolzovanie-analogovogo-komparatora.html?ysclid=m2unwqg7ti955833757

10. Микроконтроллер AVR и UART. Часть 2. Использование прерываний. [Электронный ресурс] — URL: <https://microtechnics.ru/mikrokontrolleryavr-uart-ispolzovanie-preryvanij/>

# Приложение

|  |
| --- |
| /\*  \* lab4.c  \*  \* Created: 13.12.2024 0:19:13  \* Author : Антон  \*/  #include <avr/io.h>  #include <avr/interrupt.h>  #include <util/delay.h>  #include <avr/eeprom.h>  #define F\_CPU 8000000  #define BAUD 9600L  #define UBRRL\_value (F\_CPU/(BAUD\*16))-1  #define EEPROM\_SIZE 512  volatile uint8\_t transmit\_ready = 0;  void USART\_init(){  UBRRL = UBRRL\_value;  UBRRH = (UBRRL\_value >> 51);  UCSRB = (1 << RXEN) | (1 << TXEN);  UCSRC = (1<<UCSZ0)|(1<<URSEL);  }  void USART\_sendChar(char value)  {  while (!(UCSRA & (1 << UDRE)));  UDR = value;  UCSRB |= (1 << UDRIE);  }  ISR(USART\_UDRE\_vect)  {    PORTB = (1 << PB0);  \_delay\_ms(1000);  PORTB = (0 << PB0);  \_delay\_ms(1000);  UCSRB &= ~(1 << UDRIE);    }  void EEPROM\_sendChar(char value) {  while (EECR & (1 << EEWE));  EECR |= (1 << EERIE);  eeprom\_write\_byte(0, value);  }  ISR(EE\_RDY\_vect) {  PORTB = (1 << PB1);  \_delay\_ms(1000);  PORTB = (0 << PB1);  \_delay\_ms(1000);  EECR &= ~(1 << EERIE);  }  //INT2 initialization  void INT2\_Init(void) {  MCUCSR &= ~(0 << ISC2);  GICR |= (1 << INT2);  }  ISR(INT2\_vect) {  PORTD = (1 << PD6);  }  void clear() {  uint16\_t i;  for (i = 0; i < EEPROM\_SIZE; i++) {  eeprom\_write\_byte((uint8\_t\*)i, 0xFF);  }  }  int main(void)  {  /\* Replace with your application code \*/  DDRB = (1 << PB0) | (1 << PB1);  DDRD = (1 << PD6);    USART\_init();  INT2\_Init();  clear();  sei();  while (1)  {  if (PINA & (1 << PA0)) {  USART\_sendChar('USARTUSART');  USART\_sendChar('\n');  USART\_sendChar('\r');  }  if (PINA & (1 << PA1)) {  EEPROM\_sendChar('EEPROM');  EEPROM\_sendChar('\n');  EEPROM\_sendChar('\r');  }  }  } |