Работа с внутренней памятью EEPROM

Цель: Изучить работу с энергонезависимой памятью EEPROM и реализовать работу с ней в среде Proteus.

Все микроконтроллеры семейства Mega имеют в своем составе энергонезависимую память (**EEPROM** память). Объем этой памяти колеблется от 512 байт в моделях ATmega8x до 4 Кбайт в старших моделях. **EEPROM** память расположена в своем адресном пространстве и так же, как и ОЗУ, организована линейно. Для работы с **EEPROM** памятью используются три регистра ввода/вывода: регистр адреса, регистр данных и регистр управления.

Регистр адреса

Регистр адреса **EEPROM** памяти **EEAR (EEPROM Address Register)** физически размещается в двух **PBB EEARH:EEARL,** расположенных по адресам $1F ($3F) и $1E ($3E) соответственно. В этот регистр загружается адрес ячейки, к которой будет производиться обращение. Регистр адреса доступен как для записи, так и для чтения. При этом в регистре **EEARH** задействуются только младшие разряды (количество задействованных разрядов зависит от объема **EEPROM** памяти). Незадействованные разряды регистра **EEARH** доступны только для чтения и содержат «0».

Регистр данных

Регистр данных **EEPROM** памяти **EEDR (EEPROM Data Register)** расположен по адресу $1D ($3D). При записи в этот регистр загружаются данные, которые должны быть помещены в **EEPROM**, а при чтении в этот регистр помещаются данные, считанные из **EEPROM.**

Регистр управления

Регистр управления **EEPROM** памяти **EECR (EEPROM Control Register)** расположен по адресу $1C ($3C). Этот регистр используется для управления доступом к **EEPROM** памяти. Его описание показано ниже в таблице:

Разряд	Название	Описание
7..4	-	не используются, читаются как "0"
3	EERIE	Разрешение прерывания от EEPROM. Этот разряд управляет генерацией прерывания, возникающего при завершении цикла записи в EEPROM. Если этот разряд установлен в «1», прерывания разрешены (если флаг I регистра SREG также установлен в «1»). При сброшенном разряде EEWE (см. далее в таблице) прерывание генерируется постоянно
2	EEMWE	Управление разрешением записи в EEPROM. Состояние этого разряда определяет функционирование флага разрешения записи EEWE. Если данный разряд установлен в «1», то при записи в разряд EEWE «1» происходит запись данных в EEPROM. В противном случае установка EEWE в «1» не производит никакого эффекта. После программной установки разряд EEMWE сбрасывается аппаратно через 4 машинных цикла
1	EEWE	Разрешение записи в EEPROM. При установке

		этого разряда в «1» происходит запись данных в EEPROM (если EEMWE равен «1»)
0	EERE	Разрешение чтения из EEPROM. После установки этого разряда в «1» выполняется чтение данных из EEPROM. По окончании чтения этот разряд сбрасывается аппаратно

Для записи одного байта в EEPROM необходимо:

1. Дождаться готовности EEPROM к записи данных (ждать пока не сбросится флаг EEWE регистра EECR).

2. Дождаться завершения записи во FLASH память программ (ждать пока не сбросится флаг SPMEN регистра SPMCR).

3. Загрузить байт данных в регистр EEDR, а требуемый адрес — в регистр EEAR (при необходимости).

4. Установить в «1» флаг EEMWE регистра EECR.

5. Записать в разряд EEWE регистра EECR лог. «1» в течение 4-х машинных циклов. После установки этого разряда процессор пропускает 2 машинных цикла перед выполнением следующей инструкции.

Для чтения одного байта из EEPROM необходимо:

1. Проконтролировать состояние флага EEWE. Дело в том, что пока выполняется операция записи в EEPROM память (флаг EEWE установлен), нельзя выполнять ни чтения EEPROM памяти, ни изменения регистра адреса.

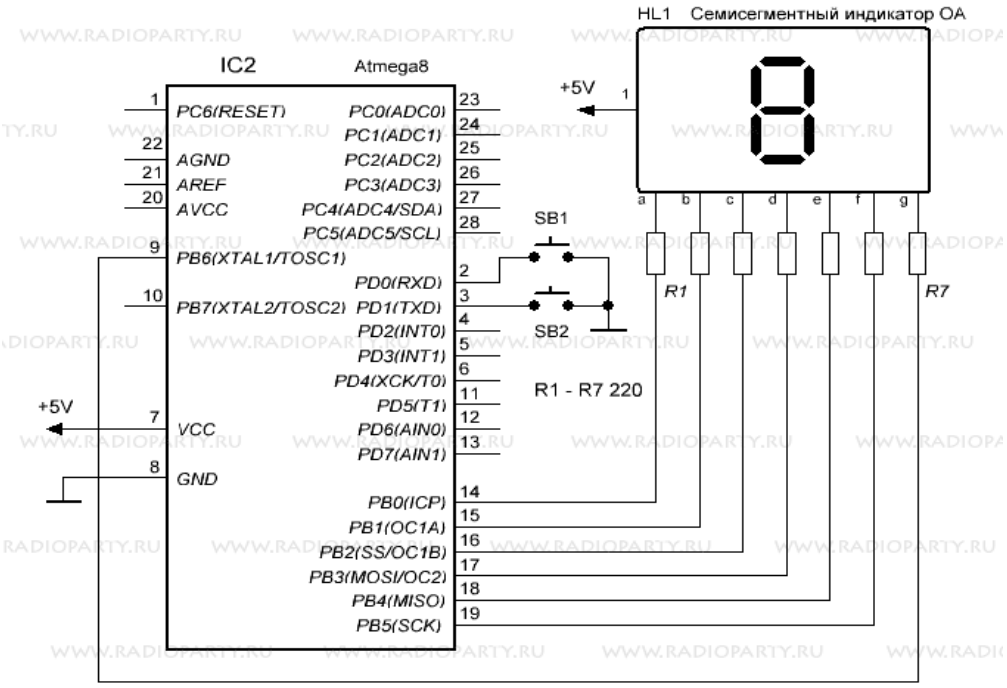
2. Загрузить требуемый адрес в регистр EEAR.

3. Установить в «1» разряд EERE регистра EECR.

Когда запрошенные данные будут помещены в регистр данных EEDR, произойдет аппаратный сброс этого разряда. Однако следить за состоянием разряда EERE для определения момента завершения операции чтения не требуется, т. к. операция чтения из EEPROM всегда выполняется за один

машинный цикл. Кроме того, после установки разряда EERE в «1» процессор пропускает 4 машинных цикла перед началом выполнения следующей инструкции.

В среде AVR Studio GCC есть стандартная библиотека для работы с EEPROM которая включается подключением файла **<avr/eeprom.h>**. Основными функциями являются **eeprom_read_byte(), eeprom_write_byte(), eeprom_read_word(), eeprom_write_word().** Для примера напишем программу мини-счетчика от 0 до 9, где при нажатии на одну кнопку будет добавляться значение, а на другую кнопку будет сохраняться это значение в памяти. Микроконтроллер Atmega8 работает от внутреннего тактового генератора частотой 8МГц. Одноразрядный семисегментный индикатор с общим анодом через токоограничительные резисторы R1-R7 подключается к порту B, общий анод к плюсу питания. Схема показана ниже:

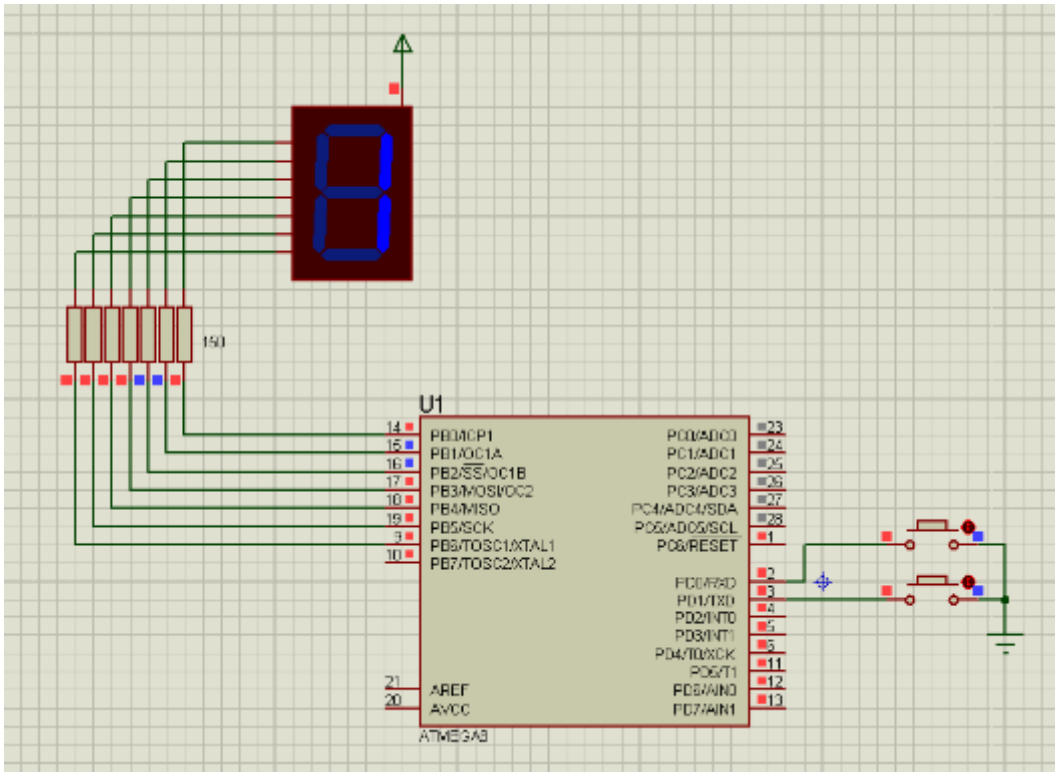


Для начала подключаем необходимые для работы библиотеки, в том числе EEPROM. Определяем переменные. Переменная "s" хранит значение для вывода на индикатор, при нажатии на кнопку SB1 это значение увеличивается на единицу, но не больше 10. Переменная eeprom_var будет взаимодействовать с EEPROM. При включении питания читается EEPROM, считанные данные присваиваются переменной "s", исходя из этого на индикатор выводится определенная цифра. При нажатии на SB2 данные из переменной "s" записываются в EEPROM, при этом индикатор мигнет один раз.

```
#include <avr/io.h>
#include <avr/eeprom.h>
#include <util/delay.h>
```

```
#define d0 ~(0x3F) // 0
#define d1 ~(0x06) // 1
#define d2 ~(0x5B) // 2
#define d3 ~(0x4F) // 3
#define d4 ~(0x66) // 4
#define d5 ~(0x6D) // 5
#define d6 ~(0x7D) // 6
#define d7 ~(0x07) // 7
#define d8 ~(0x7F) // 8
#define d9 ~(0x6F) // 9

unsigned char s;
unsigned char eeprom_var EEMEM; // определяем переменную в EEPROM

int main (void)
{
DDRB = 0xFF; // Порт B на выход
PORTB = 0xFF;
DDRD = 0x00; // Порт D на вход
PORTD = 0xFF; // Включаем подтягивающие резисторы

s = eeprom_read_byte(&eeprom_var); // считываем байт из EEPROM и помещаем его в "s"

while(1)
{
if((PIND&(1 << PD0)) == 0) // если кнопка SB1 нажата
{
while((PIND&(1 << PD0)) == 0){} // ждем отпускания кнопки
s++;                    // увеличиваем "s" на единицу
_delay_ms(200);
}

if(s == 10) // Когда дойдет до 10 обнуляем "s"
{
s = 0;
}

if((PIND&(1 << PD1)) == 0) // если кнопка SB2 нажата
{
while((PIND&(1 << PD1)) == 0){} // ждем отпускания кнопки
DDRB = 0xFF;                // мигаем индикатором
_delay_ms(200);
DDRB = 0x00;
```

```
_delay_ms(200);
DDRB = 0xFF;
eeprom_write_byte(&eeprom_var, s); // записываем "s" в EEPROM
_delay_ms(200);
}

if(s==0) // Выводим цифры на индикатор
PORTB = d0;
if(s==1)
PORTB = d1;
if(s==2)
PORTB = d2;
if(s==3)
PORTB = d3;
if(s==4)
PORTB = d4;
if(s==5)
PORTB = d5;
if(s==6)
PORTB = d6;
if(s==7)
PORTB = d7;
if(s==8)
PORTB = d8;
if(s==9)
PORTB = d9;

}
}
```



Задания для выполнения:

1. Изучить работу с энергонезависимой памятью EEPROM микроконтроллера ATMEGA8 и составить конспект.
2. Реализовать работу с памятью в среде Proteus.

Результаты и архив с проектом отправить на e-mail: rasov@rambler.ru с темой EEPROM_ФИО