Московский физико-технический институт (Государственный университет)

Введение в микроконтроллеры

Регулируемый вентилятор

Курсовой проект

Авторы: Барышников Егор Денисович, Б01-009

Михалун Дмитрий Олегович, Б01-009

Нестеренко Владислав Витальевич, Б01-009

Долгопрудный 2022

ОГЛАВЛЕНИЕ

I. ВВЕДЕНИЕ

Данный проект был подготовлен в рамках учебного курса МФТИ «Введение в микроконтроллеры».

Проект представляет собственноручно изготовленную и запрограммированную систему вентиляции, состоящую из регулируемого вентилятора и управляющего блока на базу микроконтроллера. Регулировка осуществляется с учетом температурной зависимости, введённой пользователем.

II. ЦЕЛИ И ЗАДАЧИ

Цель: создать работающую модель для вентиляции посещения с пользовательской регулировкой.

Задачи:

1. Создать рабочий прототип из микроконтроллера и вентилятора
2. Создать программу для регулировки оборотов вентилятора с учетом зависимости, введённой пользователем
3. Загрузить программу в МК
4. Провести тесты системы и исправить «баги»

III. ОПИСАНИЕ ТЕХНИЧЕСКОЙ РЕАЛИЗАЦИИ ПРОЕКТА

Прибор состоит из:

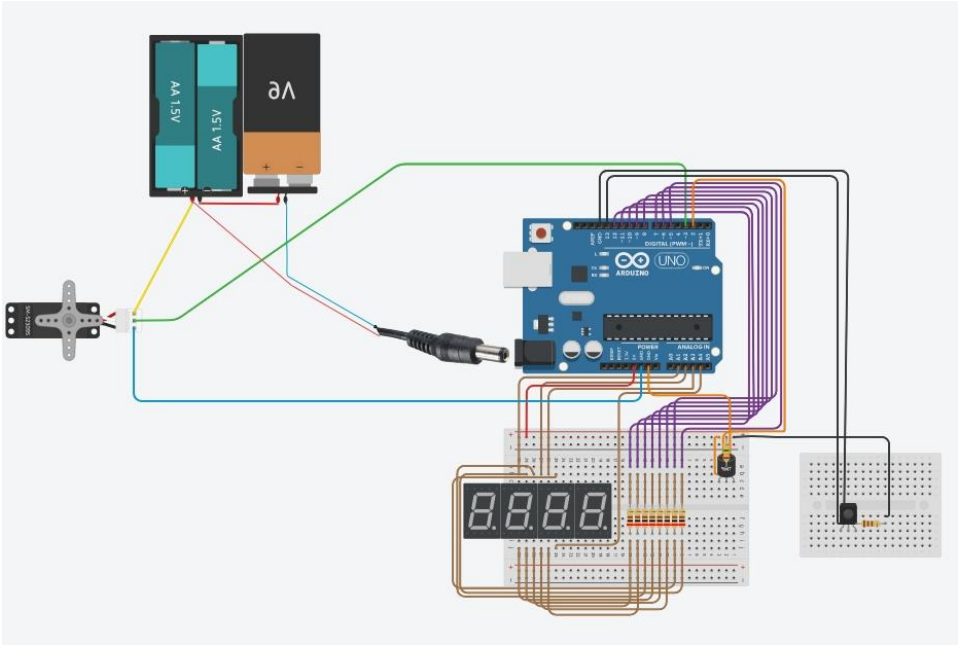
1. Вентилятор ID-COOLING RGB Series [DF-12025-ARGB]
2. Микроконтроллер Arduino Uno Atmega328-P (Частота – 16 МГц, размер программной памяти - 32 Kbytes (16K x 16), тип программной памяти – flash, размер EEPROM - 1Kx8, размер ОЗУ - 2Kx8)
3. ИК датчик - vs1838b
4. ИК пульт дистанционного управления
5. Датчик температуры Arduino DS18B20
6. Набор резисторов различных сопротивлений
7. Экран для вывода температуры
8. Комплектные провода

ИК датчик служит для получения от пользователя зависимости. Пользователь входит в меню настройки, с помощью пульта и экрана поразрядно вводит нижний и верхний пределы линейной зависимости оборотов вентилятора от температуры.

Микроконтроллер, получив зависимость (если не получил используется заводская настройка), считывает с датчика температуры данные, сопоставляет с зависимостью и получает значение для PWM. Далее данные отправляются на контроллер вентилятора.

На экране вне момента ввода зависимости отражается текущая температура.

Схема устройства представлена ниже:



IV. ОПИСАНИЕ КОДА

```
#include <IRremote.h> //библиотека для vs1838b
#include <OneWire.h> //библиотека для ds18b20
#define OK 16726215
#define UP 16718055
#define DOWN 16730805

const int IR_pin = 13; //пин подключения инфракрасного приемника
decode_results results; //переменная для хранения результата приема
IRrecv irreceiver(IR_pin); //создание объекта приемника
int anodPins[] = {A1, A2, A3, A4};
int segmentsPins[] = {5, 6, 7, 8, 9, 10, 11, 12};
int a = 2; //десятки нижней температуры
int b = 0; //единицы нижней температуры
int c = 3; //десятки верхней температуры
int d = 0; //единицы верхней температуры
int inter = 0; //флаг прерывания
OneWire ds(2); //инициализация пина для термодатчика

ISR(TIMER1_OVF_vect) //обработчик прерываний по таймеру1
{
  inter = 1;
}

void setup()
{
  Serial.begin(9600);
  for (int i = 0; i < 4; i++)
  {
    pinMode(anodPins[i], OUTPUT);
  }
  pinMode(3, OUTPUT);
  for (int i = 0; i < 8; i++)
  {
    pinMode(segmentsPins[i], OUTPUT);
  }
  irreceiver.enableIRIn(); //инициализация приемника
}
```

```
int up(int count) //обработчик кнопки Up
{
  int k; //подавемый разряд
  if (count == 0)
  {
    if (b > 8) return b;
    b++;
    k = b;
  }
  if (count == 1)
  {
    if (a > 8) return a;
    a++;
    k = a;
  }
  return k;
}

int up1(int count) //обработчик кнопки Up
{
  int k; //подавемый разряд
  if (count == 0)
  {
    if (d > 8) return d;
    d++;
    k = d;
  }
  if (count == 1)
  {
    if (c > 8) return c;
    c++;
    k = c;
  }
  return k;
}

int down(int count) //обработчик кнопки down
```

```
{
  int k; //подавемый разряд
  if (count == 0)
  {
    if (b < 1) return b;
    b--;
    k = b;
  }
  if (count == 1)
  {
    if (a < 1) return a;
    a--;
    k = a;
  }
  return k;
}

int down1(int count) //обработчик кнопки down
{
  int k; //подавемый разряд
  if (count == 0)
  {
    if (d < 1) return d;
    d--;
    k = d;
  }
  if (count == 1)
  {
    if (c < 1) return c;
    c--;
    k = c;
  }
  return k;
}

void prin(int k, int count) //вывод на индикатор ; count - разряд индикатора
{
  if (k == 0) zero(count);
```

```
  if (k == 1) one(count);
  if (k == 2) two(count);
  if (k == 3) three(count);
  if (k == 4) four(count);
  if (k == 5) five(count);
  if (k == 6) six(count);
  if (k == 7) seven(count);
  if (k == 8) eight(count);
  if (k == 9) nine(count);
}

void loop()
{
  int i;
  int k1 = b;
  int k2 = a;
  int k3 = d;
  int k4 = c;
  int counter = 0;
  if (irreceiver.decode(&results)) //пользовательский интерфейс
  {
    if (results.value == OK)
    {
      irreceiver.resume();
      while(true)
      {
       if (irreceiver.decode(&results))
       {
        if ( results.value == UP)
         k1 = up(counter);
        if ( results.value == DOWN)
          k1 = down(counter);
        if (results.value == OK)
          break;
        irreceiver.resume();
       }
       prin(k2, 1);
       prin(k1, 0);
```

```
 S(3);
}
irreceiver.resume();

  counter = 1;

  while(true)
  {
  if (irreceiver.decode(&results))
  {
   if ( results.value == UP)
    k2 = up(counter);
    if ( results.value == DOWN)
     k2 = down(counter);
   if (results.value == OK)
     break;
   irreceiver.resume();
  }
  prin(k2, 1);
  prin(k1, 0);
  S(3);
}
irreceiver.resume();

counter = 0;

while(true)
{
 if (irreceiver.decode(&results))
 {
  if ( results.value == UP)
   k3 = up1(counter);
   if ( results.value == DOWN)
    k3 = down1(counter);
  if (results.value == OK)
    break;
  irreceiver.resume();
 }
```

```
     prin(k4, 1);
     prin(k3, 0);
     S(2);
    }
   irreceiver.resume();

     counter = 1;

     while(true)
     {
     if (irreceiver.decode(&results))
     {
      if ( results.value == UP)
       k4 = up1(counter);
       if ( results.value == DOWN)
        k4 = down1(counter);
      if (results.value == OK)
        break;
      irreceiver.resume();
     }
     prin(k4, 1);
     prin(k3, 0);
     S(2);
    }
   irreceiver.resume();

  }
}
irreceiver.resume();
int num = 0; //измерение температуры
byte present = 0;
byte type_s;
byte data[12];
byte addr[8];
float celsius, fahrenheit;
int value;
if ( !ds.search(addr))
{
```

```
  ds.reset_search();
  unsigned long time = millis();
while(true)
{
  if (millis() - time > 200)
    break;
}
  return;
}
if (OneWire::crc8(addr, 7) != addr[7])
{
  return;
}
switch (addr[0])
{
  case 0x10:
    type_s = 1;
    break;
  case 0x28:
    type_s = 0;
    break;
  case 0x22:
    type_s = 0;
    break;
  default:
    return;

}
ds.reset();
ds.select(addr);
ds.write(0x44); // преобразование, используя ds.write(0x44,1) с "паразитным" питанием
unsigned long time = millis();
  // ожидание преобразования по таймеру 1
  TCNT1 = 49960;
  noInterrupts();
  TCCR1A = 0;
  TCCR1B = 0;
  TCCR1B |= (1 << CS12) | (1 << CS10);
```

```
  TIMSK1 |= (1 << TOIE1);
  interrupts();
  while(true)
  {
    if (inter == 1)
      break;
   Serial.print(" f");
   }
 noInterrupts();
 inter = 0;
 present = ds.reset();
 ds.select(addr);
 ds.write(0xBE);
 for ( i = 0; i < 9; i++)
 {
  data[i] = ds.read();
 }
 int16_t raw = (data[1] << 8) | data[0];
 if (type_s)
 {
  raw = raw << 3; // разрешение 9 бит по умолчанию
  if (data[7] == 0x10)
  {
    raw = (raw & 0xFFF0) + 12 - data[6];
  }
 }
 else
 {
  byte cfg = (data[4] & 0x60);
  if (cfg == 0x00) raw = raw & ~7; // разрешение 9 бит, 93.75 мс
    else if (cfg == 0x20) raw = raw & ~3; // разрешение 10 бит, 187.5 мс
    else if (cfg == 0x40) raw = raw & ~1;

}

 celsius = (float)raw / 16.0;
 value = (int) celsius;
 int k;
```

```
  int count = 0;
  while(value > 0) //разбиение числа на десятичные разряды
  {
    k = value%10;
    prin(k, count);
    count++;
    value = value / 10;
  }
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  if (a == c && b == d) //получение значения для PWM
    d++;
  int resulte = (int) (255 * ((int)celsius - 10*a - b) / (10*c + d - 10*a - b));
  if (resulte < 0)
    resulte = 0;
   if (resulte > 255)
    resulte = 255;
  analogWrite(3, resulte); //PWM
  TCNT1 = 49960;
    noInterrupts();
    TCCR1A = 0;
    TCCR1B = 0;
    TCCR1B |= (1 << CS12) | (1 << CS10);
    TIMSK1 |= (1 << TOIE1);
    interrupts();
    while(true)
    {
      if (inter == 1)
        break;
     Serial.print(" f");
     }
  noInterrupts();
  inter = 0;
}
                  //функции вывода цифр на индикатор
void nine(int count)
```

```
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
 int seg[] = {1, 1, 1, 1, 0, 1, 1, 0};
  for (i =0; i<= 7; i++)
  {
     if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
     else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
}
void one(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
 int seg[] = {0, 1, 1, 0, 0, 0, 0, 0};
  for (i =0; i<= 7; i++)
  {
     if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
     else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
}

void two(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
```

```
  }
  digitalWrite(anodPins[3-count], LOW);
 int seg[] = {1, 1, 0, 1, 1, 0, 1, 0};
  for (i =0; i<= 7; i++)
  {
     if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
     else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
}

void three(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
 int seg[] = {1, 1, 1, 1, 0, 0, 1, 0};
  for (i =0; i<= 7; i++)
  {
     if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
     else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
}

void four(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
   if (count == 1) digitalWrite(anodPins[0], LOW);
 int seg[] = {0, 1, 1, 0, 0, 1, 1, 0};
```

```
  for (i =0; i<= 7; i++)
  {
     if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
     else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
  digitalWrite(anodPins[0], HIGH);
}

void five(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
   if (count == 1) digitalWrite(anodPins[0], LOW);
 int seg[] = {1, 0, 1, 1, 0, 1, 1, 0};
  for (i =0; i<= 7; i++)
  {
     if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
     else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
  digitalWrite(anodPins[0], HIGH);
}

void zero(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
    digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
 int seg[] = {1, 1, 1, 1, 1, 1, 0, 0};
  for (i =0; i<= 7; i++)
```

```
  {
    if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
    else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
}

void six(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
   digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
   if (count == 1) digitalWrite(anodPins[0], LOW);
 int seg[] = {1, 0, 1, 1, 1, 1, 1, 0};
  for (i =0; i<= 7; i++)
  {
    if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
    else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
  digitalWrite(anodPins[0], HIGH);
}

void seven(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
   digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
   if (count == 1) digitalWrite(anodPins[0], LOW);
 int seg[] = {1, 1, 1, 0, 0, 0, 0, 0};
  for (i =0; i<= 7; i++)
  {
```

```
    if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
    else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
  digitalWrite(anodPins[0], HIGH);
}

void S(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
   digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
   if (count == 1) digitalWrite(anodPins[0], LOW);
 int seg[] = {0, 0, 0, 0, 0, 0, 0, 1};
  for (i =0; i<= 7; i++)
  {
    if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
    else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
  digitalWrite(anodPins[0], HIGH);
}

void eight(int count)
{
  int i;
  for (i =0; i<= 3; i++)
  {
   digitalWrite(anodPins[i], HIGH);
  }
  digitalWrite(anodPins[3-count], LOW);
   if (count == 1) digitalWrite(anodPins[0], LOW);
 int seg[] = {1, 1, 1, 1, 1, 1, 1, 0};
  for (i =0; i<= 7; i++)
  {
```

```
    if (seg[i] == 1) digitalWrite(segmentsPins[i], HIGH);
    else digitalWrite(segmentsPins[i], LOW);
  }
  delay(10);
  digitalWrite(anodPins[0], HIGH);
}
```

V. ОТВЕТЫ НА ВОПРОСЫ

1. ***Ваша фамилия, имя, отчество, номер группы?***
    a. Нестеренко Владислав Витальевич, Б01-009
    b. Барышников Егор Денисович, Б01-009
    c. Михалун Дмитрий Олегович, Б01-009

2. ***Фамилия, имя, отчество лектора?***
    Донов Геннадий Иннокентьевич

3. ***Чем отличается микроконтроллер от микропроцессора?***

Микропроцессор содержит только то, что необходимо для выполнения арифметических и логических операций: арифметико-логическое устройство(ALU), регистры, блок управления(CU). В микроконтроллер встроены программная память, DATA SEAM и дополнительные блоки, такие как: порты I/O, таймеры, АЦП и т.д.. Другими словами, у микроконтроллера основные модули, необходимые для выполнения своих функций: встроенные, а микропроцессору нужно задействовать внешние устройства.

4. ***Какие тактовые частоты могут быть у ATmega8535?***

От внутреннего генератора: 1 МГц, 2 МГц, 4 МГц, 8 МГц

От внешнего источника: частоты в диапазоне от 100 КГц до 16 МГц

5. ***Почему при повышении тактовой частоты микроконтроллера он начинает больше греться?***

Основная составляющая инвертора - конденсатор, задающий напряжение на его выходе. Энергия, затрачиваемая на зарядку конденсатора: W = CU^2/2. Энергия разрядки конденсатора через сопротивление: Wr = CU^2/2. За один такт напряжение, подаваемое на инвертор, меняется дважды. Соответственно, за один такт происходит разрядка и зарядка конденсатора. Следовательно, энергия, потребляемая за 1 такт равна W = CU^2, а мощность равна P = Wf = ϒf.

6. ***Какие таймеры есть у ATmega8535?***

2 восьмиразрядных таймера(Tim 0, Tim 2), 1 шестнадцатиразрядный(Tim 1)

7. ***Сколько режимов есть у таймера 1 и режима с каким номером у него нет?***

У таймера 1 есть 15 режимов:

1. Режим Normal (режим 0)
2. Режим CTC (режимы 4 и 12)
3. Режимы Fast PWM (режимы 5, 6, 7, 14, 15)
4. Режимы Phase Correct PWM (режимы 1, 2, 3, 10, 11)

5. Режимы Phase and Frequency Correct PWM (режимы 8, 9)

Режим под номером 13 отсутствует.

8. Внутренняя структура МК

МК состоит из блока управления питанием, блока управления сбросом, блока синхронизации, памяти программ, процессора, портов ввода-вывода, ОЗУ.

9. Какие значения записаны в TCCR после сигнала RESET?

TCCR содержит все нули.

10. Порт А. Сколько прерываний и сколько регистров ввода/вывода принадлежит порту А? Назначение этих регистров ввода/вывода?

0 прерываний, 3 регистра: PORTA (Data Register) – выходные значения, DDRA (Data Direction Register) – выбор того, на вход или выход работает вывод порта, PINA (Port A Input Pins) – значения выводов, содержимое можно только читать.

11. Регистр SREG. Назначение его разрядов?

0 – C признак переноса, 1 – Z признак нуля, 2 – N признак отрицательного результата, 3 – V признак переполнения, 4 – S бит знака, 5 – H бит переноса между полубайтами, 6 – T временное хранение бита, 7 – I бит глобального разрешения прерываний.

12. Почему после сигнала RESET все прерывания запрещены?

Чтобы правильно инициализировать работу микроконтроллера.

13. Приведите пример использования разряда T в регистре SREG.

Bst r1, 4

Bld r0, 2

14. Таймер 0. Режимы работы, количество прерываний, регистры ввода/вывода, принадлежащие таймеру 0.

Имеет 4 режима работы:

1. Normal
2. CTC
3. Phase correct PWM
4. Fast PWM

2 прерывания:

1. TIMER0_OVF
2. TIMER0_COMP

Регистры ввода-вывода:

1. TCNT0 – регистр-счетчик
2. TCCR0 – регистр контроля: 3 бита задающие частоту, 2 бита задающие режим, 2 бита, задающие значение на выходе OC0, бит FOC0.
3. Нулевой бит SFIOR – Special Function IO Register (программное обнуление делителя)
4. 2 бита TIMSK – маска прерываний
5. 2 бита TIFR – регистр флагов прерываний

15. В каких режимах таймера 0 порог изменяется не сразу (двойная буферизация записи) при записи нового значения в регистр порога с помощью команды OUT?

В ШИМ-режимах (первый и третий) для избежания glitch.

16. Откуда приходит сигнал на вход TCNTO?

С выхода управляемого предварительного делителя частоты(prescaler).

17. Как можно разрешить (запретить) прерывания по переполнению таймера 0?

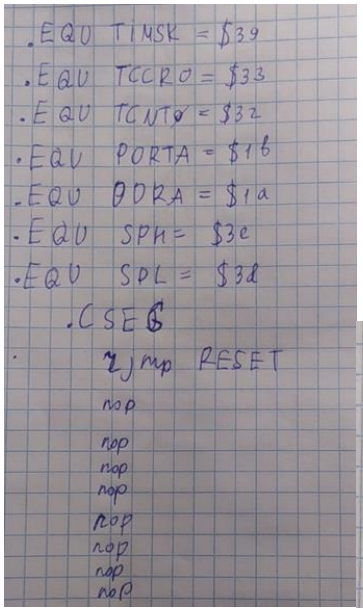
ldi r16, $01

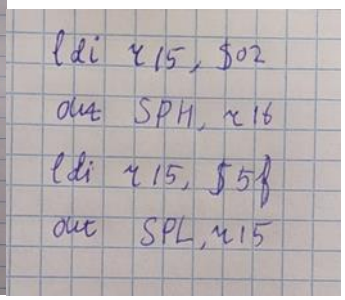
out TIMSK, r16 – разрешить

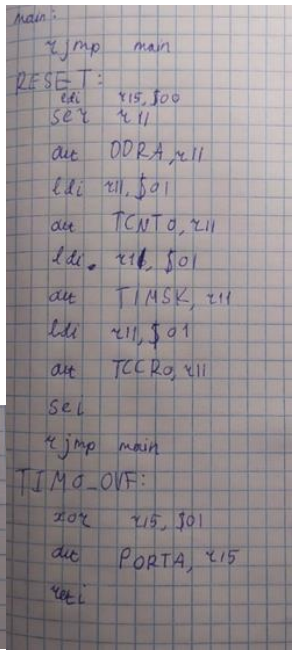
ldi r16, $00

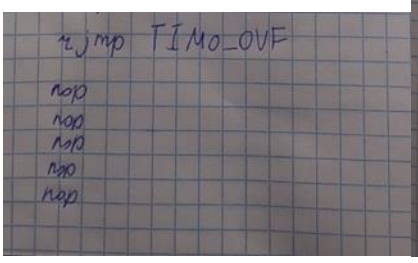
out TIMSK, r16 – запретить

18. ***<u>Напишите программу с использованием таймера 0, вырабатывающую симметричное прямоугольное колебание на одном из выходов порта A.</u>***









19. Какие коэффициенты деления частоты позволяет получать предварительный делитель таймера 0?

0.

1, 8, 64, 256, 1024

20. Какой режим таймера 0 позволяет вырабатывать треугольные колебания, используя дополнительную интегрирующую цепочку?

Любой: в не-ШИМ режимах достаточно поставить OC0 изменяться при совпадении с порогом; а в ШИМ режимах достаточно выставить порог 0.5 от максимального значения (скважность 0.5).

21. Как запрограммировать предварительный делитель таймера 0?

Выставить в биты 2:0 регистра TCCR0 значение от 1 до 5.

22. Режим 0 таймера 0.

Режим Normal – счетчик TCNT0 увеличивается на 1 по каждому импульсу тактового сигнала. При переходе через $FF «1» записывается в бит TOV0 регистра TIFR, счетчик обнуляется. При совпадении содержимого OCR0 и TCNT0 записывается «1» в бит OCF0 регистра TIFR.

23. *Режим 1 таймера 0*.

Режим Phase Correct PWM – ШИМ с точной фазой – генерация сигналов с широтно-импульсной модуляцией. Работает на сложение от $00 до $FF, затем на вычитание обратно до $00, затем происходит прерывание и смена направления счетчика. При совпадении содержимого счетчика с порогом изменяется состояние выхода OC0. Особенность этого режима – двойная буферизация записи в регистр порога OCR0 – новое значение сохраняется в буферном регистре, а значение OCR0 изменяется только после прохождения $FF.

24. *Режим 2 таймера 0*.

Режим CTC – Clear Timer on Compare Match – режим счета по модулю, который определяется содержимым регистра порога. Происходит прибавление до совпадения значения счетчика с содержимым OCR0, затем прерывание по сравнению, счетчик переходит в состояние $00 и процесс повторяется.

25. *Режим 3 таймера 0*.

Режим Fast PWM используется для генерации высокочастотного сигнала с широтно-импульсной модуляцией. Состояние TCNT0 меняется от $00 до $FF, затем он обнуляется и процесс повторяется. Также есть двойная буферизация, при совпадении содержимого счетчика с пороговым изменяется состояние выхода OC0.

26. Когда меняется порог в режиме 3 таймера 0?

Изменение значения регистра происходит после достижения счетчиком значения $FF, до этого новое значение находится в буферном регистре.

27. Можно ли писать в TCNT0 без остановки счета?

Можно, но есть риск пропустить прерывание.

28. Как можно остановить счет в таймере 0?

Обнулить биты 2:0 регистра TCCRO.

29. *Система прерываний микроконтроллера ATmega8535*.

Система из 21 прерывания, чем меньше номер прерывания, тем оно приоритетнее. При генерации разрешенного прерывания все прерывания запрещаются глобально и исполнение переходит в вектор прерывания. При выходе командой reti восстанавливается ход исполнения и включаются глобально прерывания.

30. Сколько всего прерываний у ATmega8535?

21

31. Как организовать вложенные прерывания?

Нужно разрешить глобальные прерывания в подпрограмме прерывания.

32. Как можно разрешить (запретить) одновременно все прерывания?

Ассемблерные команды: sei – разрешить, cli – запретить.

33. Как организована система приоритетов при обработке прерываний?

Каждому прерыванию присваивается номер и первым обрабатывается прерывание с наименьшим номером.

34. Какое минимальное время требуется для преобразования в АЦП?

65 мкс

35. Чем сигнальный процессор отличается от МК?

У микроконтроллера основная задача – это работа с периферийными устройствами, а сигнальный процессор имеет специфичный набор команд и регистров, чтобы быстрее обрабатывать сигналы.

36. Зачем в программе надо устанавливать начальное значение Stack Pointer и чему это значение должно быть равно?

После RESET не гарантируется правильное значение SP. После RESET не гарантируется правильное значение SP. Желательно $25F, но не меньше $60.

37. *Сторожевой таймер и особенности его работы*.

WatchDog Timer предназначен для ликвидаций последствий сбоев в работе микроконтроллера. Он через определенный промежуток времени выполняет RESET.

Чтобы сторожевой таймер не срабатывал при правильной работе программы, используется команда WDR для его сброса.

38. Что такое SPI и зачем он нужен?

Это последовательный синхронный интерфейс, который позволяет передавать данные с высокой скоростью между ATmega8535 и внешними устройствами.

39. Как инициировать передачу байта в SPI?

Нужно записать байт в регистр SPDR у MASTER – ведущего микроконтроллера, тогда произойдет передача между ним и SLAVE – ведомым микроконтроллером.

40. Сколько прерываний и сколько регистров ввода/вывода принадлежит SPI?

a. 1 прерывание: SPI_STC
b. 3 регистра ввода/вывода: SPCR, SPSR, SPDR.

41. Сколько проводов необходимо для реализации однопроводного интерфейса?

Для 1 проводной шины необходимы 2 провода – сигнальный, подключенный к плюсу, и провод, подключенный к общему проводу (минусу).

42. Как выглядит физический ноль и физическая единица?

a. 0 – низкое напряжение (ниже некоторого порогового значения)
b. 1 – высокое напряжение (выше некоторого порогового значения)

43. Как в однопроводном интерфейсе передается информационный ноль, информационная единица? Какова максимальная скорость передачи?

Логически. Подается физический ноль длительностью 1-15 мкс, затем физический ноль или физическая единица длительностью до 60 мкс, следовательно, считывается 0 или 1.

44. Что такое серийный номер в однопроводном интерфейсе и какова его структура?

Серийный номер является идентификатором устройства; его структура: 64 бита, из которых 8 бит - код семейства, 48 бит - серийный номер, 8 бит - контрольная сумма.

45. Какая команда позволяет Master определить номера всех Slave в сети MicroLAN?

Search ROM

46. Как выглядит сигнал сброса в сети MicroLAN?

Долгий импульс низкого уровня продолжительностью минимум 480мкс, затем долгий импульс высокого уровня, тоже минимум 480мкс.

VI. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Донов Г.И. «Применение микроконтроллеров», издание 2007 года
2. https://github.com/Gugush284/MC_fan.git
3. http://codius.ru/articles/Arduino_UNO_4разрядный_7сегментный_идикатор_12_pin_3641BS_red
4. https://arduinomaster.ru/datchiki-arduino/arduino-ds18b20/