Практическое занятие №
Тема: «Подсистема питания в микроконтроллерных системах»

Цель работы: приобрести практические навыки по использованию подсистемы питания в микроконтроллерных системах на платфоре Arduino Uno R3.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно заданию в примере.

Содержание отчета:

– название практического занятия, его цель;
– фото или скриншоты собранной схемы;
– написанный программный код вставить текстом;
– вывод о проделанной работе;
– файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Режимы питания и сна

Микроконтроллер ATmega328P поддерживает несколько режимов пониженного энергопотребления:

1. **Idle Mode** - отключается CPU, но работают периферийные устройства.
2. **ADC Noise Reduction** - отключается CPU и АЦП.
3. **Power-down** - полное отключение почти всех систем.
4. **Power-save** - аналогичен Power-down, но с дополнительными опциями
5. **Standby** - отключается CPU, но оставляется тактовый генератор

Источники пробуждения

- Внешние прерывания (INT0, INT1);
- Прерывания по изменению состояния пинов (PCINT);
- Срабатывание watchdog таймера;
- Аналоговый компаратор;
- Другие периферийные устройства.

ЗАДАНИЕ

Программа 1: Базовые режимы сна

```
#include <avr/sleep.h>
#include <avr/power.h>

const int ledPin = 13;
const int wakeButton = 2;  // INT0

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(wakeButton, INPUT_PULLUP);
  Serial.begin(9600);

  // Настройка прерывания для пробуждения
  attachInterrupt(digitalPinToInterrupt(wakeButton), wakeUp,
LOW);
}

void loop() {
  Serial.println("Выберите режим сна:");
  Serial.println("1 - Idle Mode");
  Serial.println("2 - ADC Noise Reduction");
  Serial.println("3 - Power-down");
  Serial.println("4 - Power-save");
  Serial.println("5 - Standby");

  while (!Serial.available()) {
    delay(100);
  }

  int mode = Serial.read() - '0';

  switch (mode) {
    case 1:
      enterIdleMode();
      break;
    case 2:
      enterADCMode();
      break;
    case 3:
      enterPowerDown();
      break;
    case 4:
```

```
      enterPowerSave();
      break;
    case 5:
      enterStandby();
      break;
    default:
      Serial.println("Неверный выбор");
      return;
  }

  // После пробуждения
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
  delay(500);
  digitalWrite(ledPin, HIGH);
  delay(500);
  digitalWrite(ledPin, LOW);
}

void enterIdleMode() {
  Serial.println("Вход в Idle Mode...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_IDLE);
  sleep_mode();
}

void enterADCMode() {
  Serial.println("Вход в ADC Noise Reduction...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_ADC);
  sleep_mode();
}

void enterPowerDown() {
  Serial.println("Вход в Power-down...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_mode();
}

void enterPowerSave() {
  Serial.println("Вход в Power-save...");
  delay(100);
```

```
  set_sleep_mode(SLEEP_MODE_PWR_SAVE);
  sleep_mode();
}

void enterStandby() {
  Serial.println("Вход в Standby...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_STANDBY);
  sleep_mode();
}

void wakeUp() {
  // Пустая функция, просто для пробуждения
}
```

Программа 2: Пробуждение по различным источникам

```
#include <avr/sleep.h>
#include <avr/power.h>
#include <avr/wdt.h>

const int ledPin = 13;
const int button1 = 2;  // INT0
const int button2 = 3;  // INT1
const int button3 = 4;  // PCINT

volatile bool wokeByWatchdog = false;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(button1, INPUT_PULLUP);
  pinMode(button2, INPUT_PULLUP);
  pinMode(button3, INPUT_PULLUP);

  Serial.begin(9600);

  // Настройка прерываний
  attachInterrupt(digitalPinToInterrupt(button1),
wakeUpExt0, FALLING);
  attachInterrupt(digitalPinToInterrupt(button2),
wakeUpExt1, FALLING);

  // Настройка PCINT для кнопки 3
  PCICR |= (1 << PCIE2);    // Включить PCINT для порта D
```

```
  PCMSK2 |= (1 << PCINT20); // Включить PCINT для пина 4
}

void loop() {
  Serial.println("Выберите источник пробуждения:");
  Serial.println("1 - Внешнее прерывание INT0");
  Serial.println("2 - Внешнее прерывание INT1");
  Serial.println("3 - Прерывание по изменению состояния
(PCINT)");
  Serial.println("4 - Watchdog таймер (2 сек)");

  while (!Serial.available()) {
    delay(100);
  }

  int choice = Serial.read() - '0';

  switch (choice) {
    case 1:
      sleepWithExt0();
      break;
    case 2:
      sleepWithExt1();
      break;
    case 3:
      sleepWithPCINT();
      break;
    case 4:
      sleepWithWatchdog();
      break;
    default:
      Serial.println("Неверный выбор");
      return;
  }

  // Индикация пробуждения
  indicateWakeup();
}

void sleepWithExt0() {
  Serial.println("Сон с пробуждением по INT0...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
```

```
  sleep_mode();
  sleep_disable();
}

void sleepWithExt1() {
  Serial.println("Сон с пробуждением по INT1...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  sleep_mode();
  sleep_disable();
}

void sleepWithPCINT() {
  Serial.println("Сон с пробуждением по PCINT...");
  delay(100);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  sleep_mode();
  sleep_disable();
}

void sleepWithWatchdog() {
  Serial.println("Сон с пробуждением по Watchdog...");
  delay(100);

  // Настройка watchdog таймера
  wdt_enable(WDTO_2S);
  WDTCSR |= (1 << WDIE);

  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  sleep_mode();
  sleep_disable();

  wdt_disable();
}

void indicateWakeup() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(ledPin, HIGH);
    delay(200);
    digitalWrite(ledPin, LOW);
    delay(200);
```

```
  }
}

// Обработчики прерываний
void wakeUpExt0() {
  // Пробуждение по INT0
}

void wakeUpExt1() {
  // Пробуждение по INT1
}

ISR(PCINT2_vect) {
  // Пробуждение по PCINT
}

ISR(WDT_vect) {
  // Обработчик прерывания watchdog таймера
  wokeByWatchdog = true;
}
```

Программа 3: Комплексная система с множеством режимов

```
#include <avr/sleep.h>
#include <avr/power.h>
#include <avr/wdt.h>

// Пины
const int ledPin = 13;
const int modeButton = 2;    // INT0
const int wakeButton = 3;    // INT1
const int sensorPin = A0;

// Переменные
volatile int sleepMode = 0;
volatile bool shouldSleep = false;
volatile bool wokeByButton = false;
unsigned long lastActivity = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(modeButton, INPUT_PULLUP);
  pinMode(wakeButton, INPUT_PULLUP);

  Serial.begin(9600);
```

```
  // Настройка прерываний
  attachInterrupt(digitalPinToInterrupt(modeButton),
changeMode, FALLING);
  attachInterrupt(digitalPinToInterrupt(wakeButton),
wakeUpNow, FALLING);

  Serial.println("Система управления питанием Arduino");
  printMenu();
}

void loop() {
  // Проверка активности - автосон через 30 секунд
  if (millis() - lastActivity > 30000) {
    Serial.println("Автоматический переход в сон...");
    enterSelectedSleepMode();
  }

  // Обработка команд
  if (Serial.available()) {
    handleCommand(Serial.read());
    lastActivity = millis();
  }

  // Режим измерения датчика
  if (sleepMode == 0) {
    readSensor();
    delay(1000);
  }

  // Принудительный сон
  if (shouldSleep) {
    shouldSleep = false;
    enterSelectedSleepMode();
  }
}

void handleCommand(char cmd) {
  switch (cmd) {
    case '1':
      sleepMode = 0;
      Serial.println("Режим: Активный (измерение датчика)");
      break;
    case '2':
```

```
      sleepMode = 1;
      Serial.println("Режим: Idle (экономный)");
      break;
    case '3':
      sleepMode = 2;
      Serial.println("Режим: Power-down (глубокий сон)");
      break;
    case '4':
      shouldSleep = true;
      break;
    case '5':
      printStatus();
      break;
    case '?':
      printMenu();
      break;
    default:
      Serial.println("Неизвестная команда");
  }
}

void enterSelectedSleepMode() {
  Serial.println("Переход в режим сна...");
  digitalWrite(ledPin, LOW);
  delay(100);

  switch (sleepMode) {
    case 0:
      // Активный режим - не спим
      break;
    case 1:
      set_sleep_mode(SLEEP_MODE_IDLE);
      sleep_mode();
      break;
    case 2:
      set_sleep_mode(SLEEP_MODE_PWR_DOWN);
      sleep_enable();
      sleep_mode();
      sleep_disable();
      break;
  }

  // После пробуждения
  digitalWrite(ledPin, HIGH);
```

```
  Serial.println("Пробуждение!");
  lastActivity = millis();
}

void readSensor() {
  int sensorValue = analogRead(sensorPin);
  float voltage = sensorValue * (5.0 / 1023.0);
  Serial.print("Значение датчика: ");
  Serial.print(sensorValue);
  Serial.print(" (");
  Serial.print(voltage);
  Serial.println("V)");
}

void printMenu() {
  Serial.println();
  Serial.println("=== Меню управления ===");
  Serial.println("1 - Активный режим (измерения)");
  Serial.println("2 - Экономный режим (Idle)");
  Serial.println("3 - Глубокий сон (Power-down)");
  Serial.println("4 - Принудительный сон");
  Serial.println("5 - Статус системы");
  Serial.println("? - Это меню");
  Serial.println();
}

void printStatus() {
  Serial.println("=== Статус системы ===");
  Serial.print("Текущий режим: ");
  switch (sleepMode) {
    case 0: Serial.println("Активный"); break;
    case 1: Serial.println("Idle"); break;
    case 2: Serial.println("Power-down"); break;
  }
  Serial.print("Время работы: ");
  Serial.print(millis() / 1000);
  Serial.println(" сек");
  Serial.println("======================");
}

// Обработчики прерываний
void changeMode() {
  static unsigned long lastInterrupt = 0;
```

```
  if (millis() - lastInterrupt > 200) { // Защита от
дребезга
    sleepMode = (sleepMode + 1) % 3;
    Serial.print("Смена режима на: ");
    switch (sleepMode) {
      case 0: Serial.println("Активный"); break;
      case 1: Serial.println("Idle"); break;
      case 2: Serial.println("Power-down"); break;
    }
    lastInterrupt = millis();
  }
}

void wakeUpNow() {
  wokeByButton = true;
}
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие основные режимы пониженного энергопотребления поддерживает ATmega328P и чем они отличаются?
2. Каковы основные источники пробуждения микроконтроллера из спящих режимов?
3. Какой режим сна наиболее экономичен по энергопотреблению и почему?
4. Что происходит с периферийными устройствами (таймеры, АЦП, USART) в различных режимах сна?
5. Каковы преимущества и недостатки использования watchdog таймера для пробуждения?
6. Как настроить микроконтроллер для пробуждения по внешнему прерыванию?
7. Какие меры предосторожности необходимы при использовании прерываний для пробуждения?
8. Как организовать периодическое пробуждение для выполнения измерений с минимальным энергопотреблением?
9. Каким образом можно уменьшить энергопотребление в активном режиме?
10. Как реализовать систему с несколькими источниками пробуждения и приоритетами?
11. В каких практических применениях критически важно использование режимов пониженного энергопотребления?
12. Как можно комбинировать различные режимы сна для оптимизации энергопотребления в сложных системах?

13. Какие факторы, кроме режима сна микроконтроллера, влияют на общее энергопотребление системы?

14. Как организовать сохранение состояния системы перед переходом в глубокий сон?

15. Какие методы отладки можно использовать при работе со спящими режимами микроконтроллера?