Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»
(ФГБОУ ВО «СамГТУ»)

Институт «Автоматики и инженерных технологий»

Работа с портами микроконтроллера и битами

Лабораторная работа №1, отчёт

Выполнил студенты

3 курса, 3-ИАИТ-110 группы

Санталов Александр
Шмарин Илья
Ляпин Павел

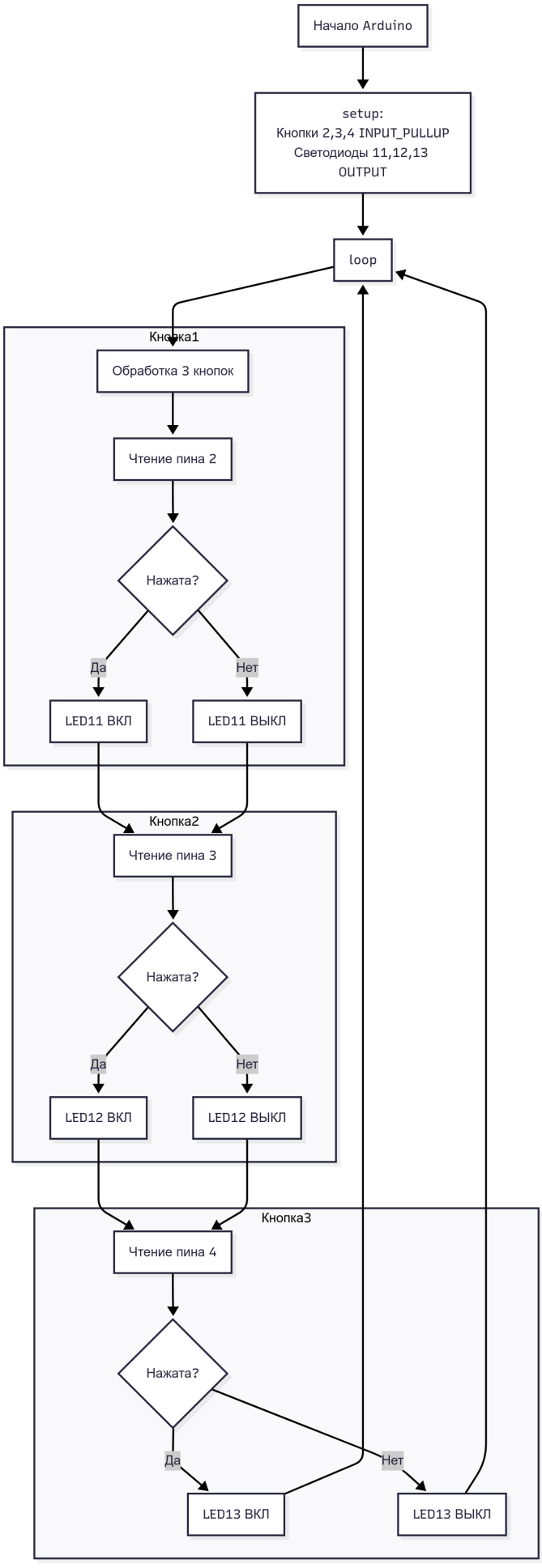
Самара, 2025 г.

Цель работы – показать работу с регистрами портов микроконтроллера Arduino (ATmega328P) для чтения и управления выводами на уровне байта и отдельных битов. Познакомиться со способами чтения всего порта целиком и методами определения, в какой позиции считанного байта установлена логическая единица или ноль, а также способами установки единицы в нужной позиции порта.

Задание 1:

Подключить к контроллеру Arduino три светодиода и три кнопки.

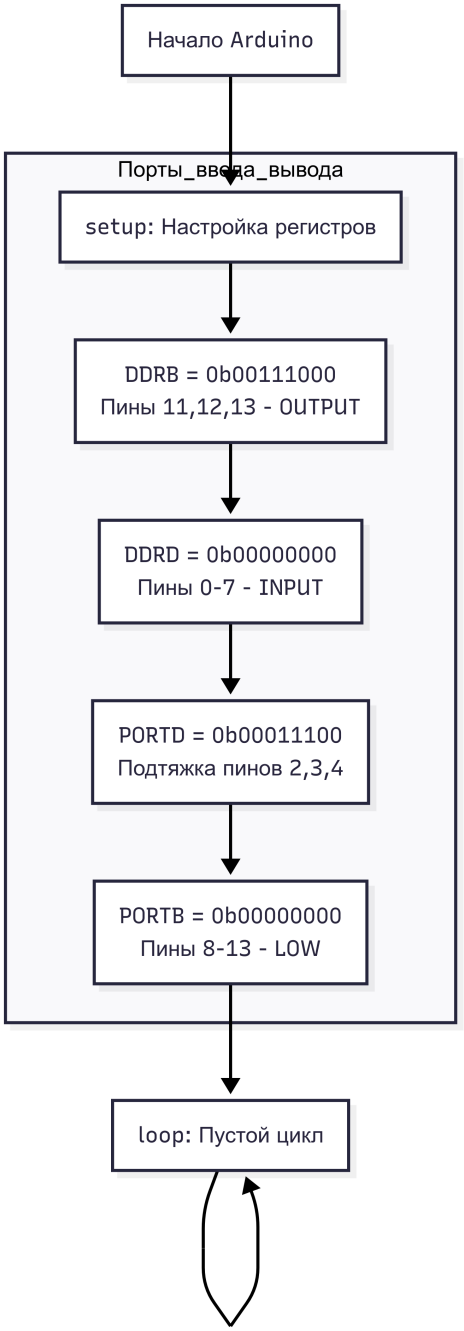
```
int buttonState_1 = 0;
int buttonState_2 = 0;
int buttonState_3 = 0;

void setup() {
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  buttonState_1 = digitalRead(2);
  if (buttonState_1 == LOW) {
  digitalWrite(11, HIGH);
  }
  else{
    digitalWrite(11, LOW);
  }
  buttonState_2 = digitalRead(3);
  if (buttonState_2 == LOW) {
  digitalWrite(12, HIGH);
  }
  else{
    digitalWrite(12, LOW);
  }
  buttonState_3 = digitalRead(4);
  if (buttonState_3 == LOW) {
  digitalWrite(13, HIGH);
  }
  else{
    digitalWrite(13, LOW);
  }
}
```


Начало Arduino
setup:
Кнопки 2,3,4 INPUT_PULLUP
Светодиоды 11,12,13
OUTPUT
loop
Кнопка1
Обработка 3 кнопок
Чтение пина 2
Нажата?
Да
Нет
LED11 ВКЛ
LED11 ВЫКЛ
Кнопка2
Чтение пина 3
Нажата?
Да
Нет
LED12 ВКЛ
LED12 ВЫКЛ
Кнопка3
Чтение пина 4
Нажата?
Да
Нет
LED13 ВКЛ
LED13 ВЫКЛ

Задание 2.1:

Настроить соответствующие биты DDRx: часть портов — входы с подтяжкой, часть — дискретные выходы для светодиодов.

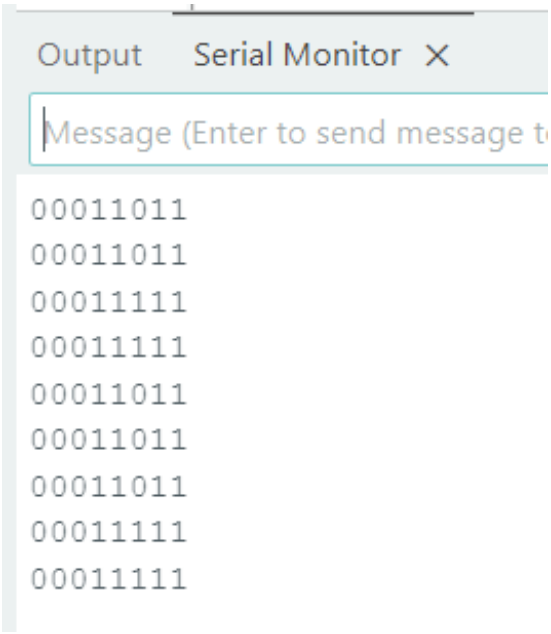
```
void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
  PORTB = 0b00000000;
}
```

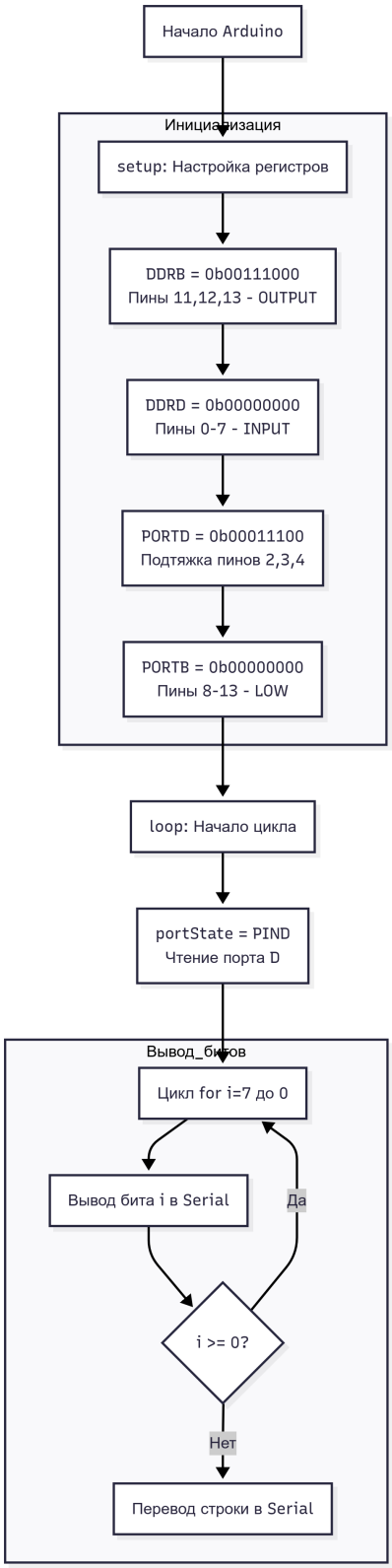


Задание 2.2:

Программно прочитать байт состояния входного порта и вывести его в терминал (в бинарном виде).

```
void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
  PORTB = 0b00000000;
  Serial.begin(9600);
}

void loop() {
  uint8_t portState = PIND;

  for (int i = 7; i >= 0; i--) {
    Serial.print((portState >> i) & 1);
  }
  Serial.println();

  delay(200);
}
```



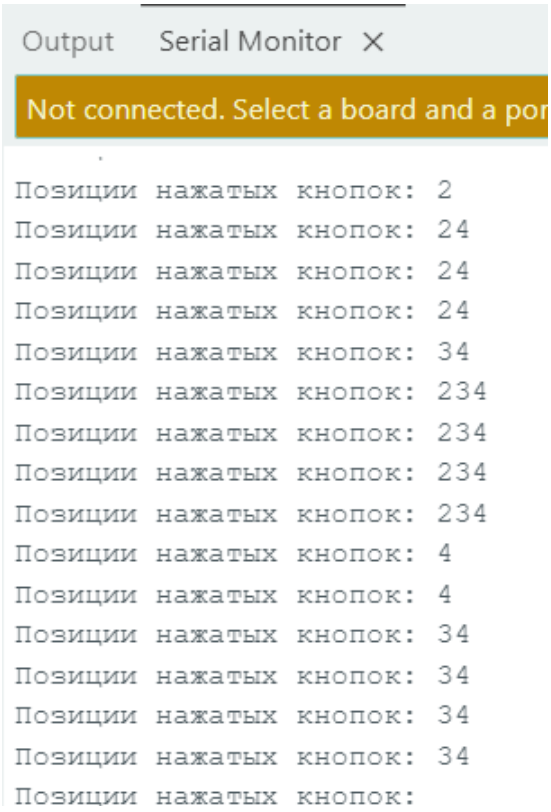


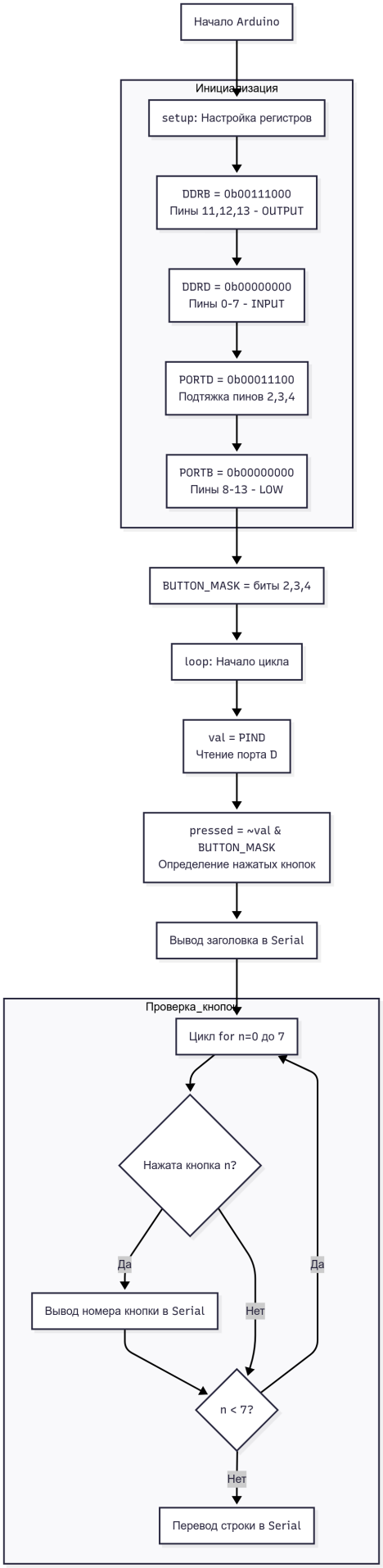
Задание 2.3:

С помощью битовых операций определить какие кнопки нажаты и вывести в терминал номера позиций установленных единиц.

```
void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
```

```
  PORTB = 0b00000000;
  Serial.begin(9600);
}
const uint8_t BUTTON_MASK = (1 << PD2) | (1 << PD3) | (1 << PD4);
void loop() {
  uint8_t val = PIND;
  uint8_t pressed = ~val & BUTTON_MASK;
  Serial.print("Позиции нажатых кнопок: ");
  for (uint8_t n = 0; n < 8; ++n) {
  if (pressed & (1 << n)) Serial.print(n, " ");
  }
  Serial.println();
  delay(500);
}
```



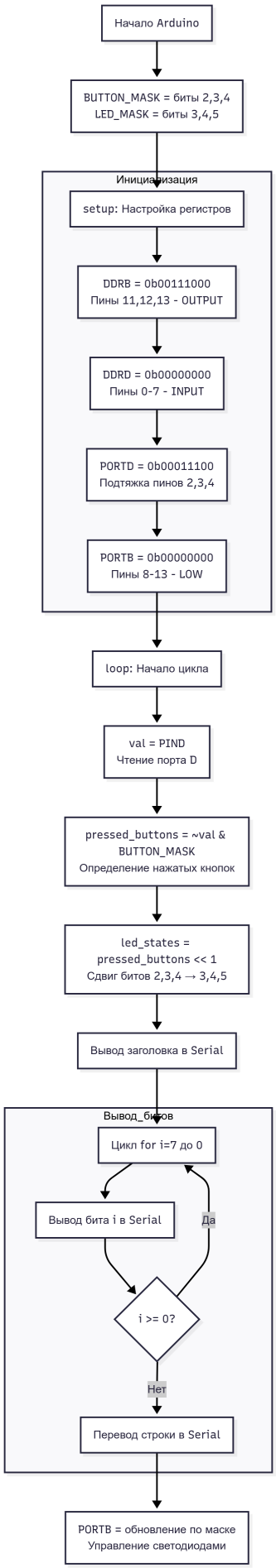

Начало Arduino
Инициализация
setup: Настройка регистров
DDRB = 0b00111000
Пины 11,12,13 - OUTPUT
DDRD = 0b00000000
Пины 0-7 - INPUT
PORTD = 0b00011100
Подтяжка пинов 2,3,4
PORTB = 0b00000000
Пины 8-13 - LOW
BUTTON_MASK = биты 2,3,4
loop: Начало цикла
val = PIND
Чтение порта D
pressed = ~val &
BUTTON_MASK
Определение нажатых кнопок
Вывод заголовка в Serial
Проверка_кнопок
Цикл for n=0 до 7
Нажата кнопка n?
Да
Вывод номера кнопки в Serial
Нет
Да
n < 7?
Нет
Перевод строки в Serial

Задание 2.4:

Сформировать байт, содержащий логические единицы в позициях нажатых кнопок, и записать этот байт одновременно в терминал и в выходной порт (при записи не менять состояние других битов порта). Для установки битов реализовать не менее двух различных способов (например чекрез маску и через bitSet/PORTx I=)

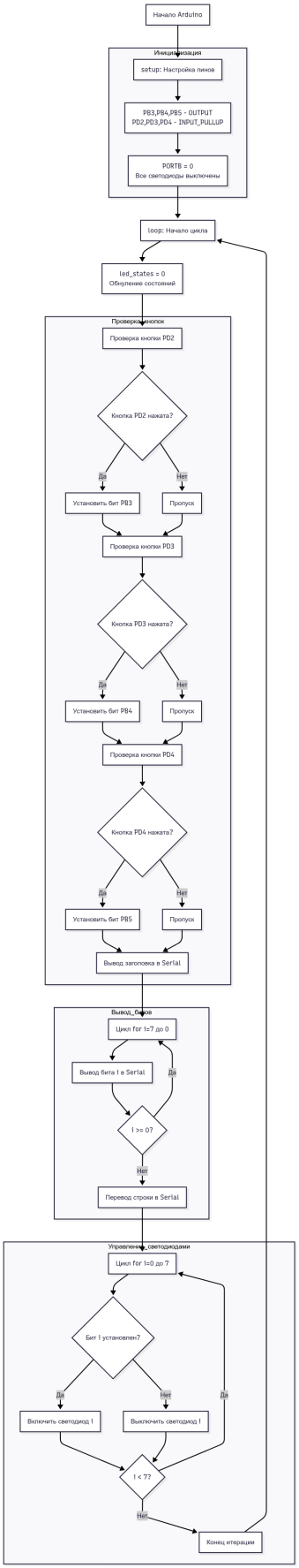
Способ 1: через битовые маски

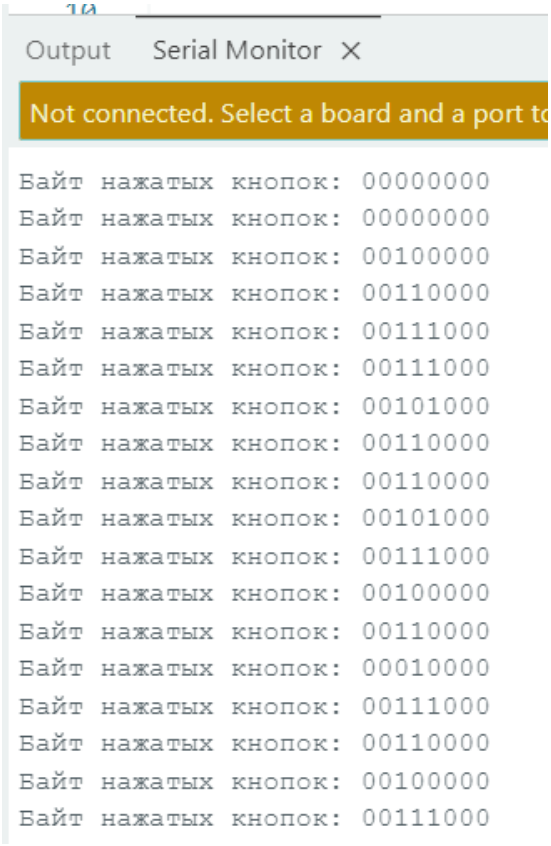
```
const uint8_t BUTTON_MASK = (1 << PD2) | (1 << PD3) | (1 << PD4);
const uint8_t LED_MASK = (1 << PB3) | (1 << PB4) | (1 << PB5);

void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
  PORTB = 0b00000000;
  Serial.begin(9600);
}

void loop() {
  uint8_t val = PIND;
  uint8_t pressed_buttons = ~val & BUTTON_MASK;
  uint8_t led_states = pressed_buttons << 1;  // Сдвиг битов 2,3,4
     3,4,5

  Serial.print("Байт нажатых кнопок: ");
  for (int i = 7; i >= 0; i--) {
    Serial.print((led_states >> i) & 1);
  }
  Serial.println();

  // Запись в порт через маску
  PORTB = (PORTB & ~LED_MASK) | (led_states & LED_MASK);

  delay(200);
}
```


Начало Arduino
BUTTON_MASK = биты 2,3,4
LED_MASK = биты 3,4,5
Инициализация
setup: Настройка регистров
DDRB = 0b00111000
Пины 11,12,13 - OUTPUT
DDRD = 0b00000000
Пины 0-7 - INPUT
PORTD = 0b00011100
Подтяжка пинов 2,3,4
PORTB = 0b00000000
Пины 8-13 - LOW
loop: Начало цикла
val = PIND
Чтение порта D
pressed_buttons = ~val &
BUTTON_MASK
Определение нажатых кнопок
led_states =
pressed_buttons << 1
Сдвиг битов 2,3,4 → 3,4,5
Вывод заголовка в Serial
Вывод_битов
Цикл for i=7 до 0
Вывод бита i в Serial
Да
i >= 0?
Нет
Перевод строки в Serial
PORTB = обновление по маске
Управление светодиодами

Способ 2: через bitSet/bitClear

```
void setup() {
  bitSet(DDRB, PB3); bitSet(DDRB, PB4); bitSet(DDRB, PB5);
  bitClear(DDRD, PD2); bitClear(DDRD, PD3); bitClear(DDRD, PD4);
  bitSet(PORTD, PD2); bitSet(PORTD, PD3); bitSet(PORTD, PD4);
  PORTB = 0;
  Serial.begin(9600);
}

void loop() {
  uint8_t led_states = 0;

  if (!bitRead(PIND, PD2)) bitSet(led_states, PB3);  // Кнопка 2     LED 11
  if (!bitRead(PIND, PD3)) bitSet(led_states, PB4);  // Кнопка 3     LED 12
  if (!bitRead(PIND, PD4)) bitSet(led_states, PB5);  // Кнопка 4     LED 13

  Serial.print("Байт нажатых кнопок: ");
  for (int i = 7; i >= 0; i--) {
    Serial.print(bitRead(led_states, i));
  }
  Serial.println();

  for (int i = 0; i < 8; i++) {
    if (bitRead(led_states, i)) {
      bitSet(PORTB, i);
    } else {
      bitClear(PORTB, i);
    }
  }

  delay(200);
}
```


Начало Arduino
Инициализация
setup: Настройка пинов
PB3,PB4,PB5 - OUTPUT
PD2,PD3,PD4 - INPUT_PULLUP
PORTB = 0
Все светодиоды выключены
loop: Начало цикла
led_states = 0
Обнуление состояний
Проверка кнопок
Проверка кнопки PD2
Кнопка PD2 нажата?
Да
Нет
Установить бит PB3
Пропуск
Проверка кнопки PD3
Кнопка PD3 нажата?
Да
Нет
Установить бит PB4
Пропуск
Проверка кнопки PD4
Кнопка PD4 нажата?
Да
Нет
Установить бит PB5
Пропуск
Вывод заголовка в Serial
Вывод_битов
Цикл for i=7 до 0
Вывод бита i в Serial
Да
i >= 0?
Нет
Перевод строки в Serial
Управление светодиодами
Цикл for i=0 до 7
Бит i установлен?
Да
Нет
Включить светодиод i
Выключить светодиод i
Да
i < 7?
Нет
Конец итерации



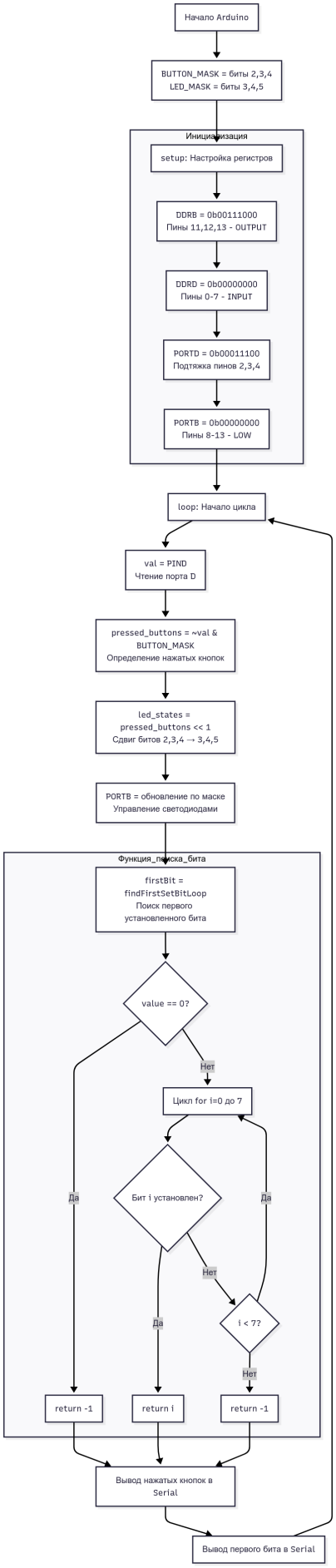
Задание 2.5:

Реализовать функцию, возвращающую индекс (позицию) первого установленного бита (по младшему или старшему разряду — на выбор) и продемонстрировать её работу. В решении показать реализацию через цикл и через __builtin_ctz/__builtin_ffs.

Способ 1: через цикл

```
const uint8_t BUTTON_MASK = (1 << PD2) | (1 << PD3) | (1 << PD4);
const uint8_t LED_MASK = (1 << PB3) | (1 << PB4) | (1 << PB5);

int findFirstSetBitLoop(uint8_t value) {
  if (value == 0) return -1;  // Ни один бит не установлен

  for (int i = 0; i < 8; i++) {
    if (value & (1 << i)) {
      return i;  // Нашли первый установленный бит (с младшего)
    }
  }
  return -1;
}

void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
  PORTB = 0b00000000;
  Serial.begin(9600);
```

```
}

void loop() {
  uint8_t val = PIND;
  uint8_t pressed_buttons = ~val & BUTTON_MASK;
  uint8_t led_states = pressed_buttons << 1;

  PORTB = (PORTB & ~LED_MASK) | (led_states & LED_MASK);

  int firstBit = findFirstSetBitLoop(pressed_buttons);

  Serial.print("Нажатые кнопки: ");
  for (int i = 7; i >= 0; i--) Serial.print((pressed_buttons >> i) & 1);
  Serial.print(" | Первый бит (цикл): ");
  if (firstBit != -1) {
    Serial.print("D");
    Serial.print(firstBit);
  } else {
    Serial.print("нет");
  }
  Serial.println();

  delay(500);
}
```

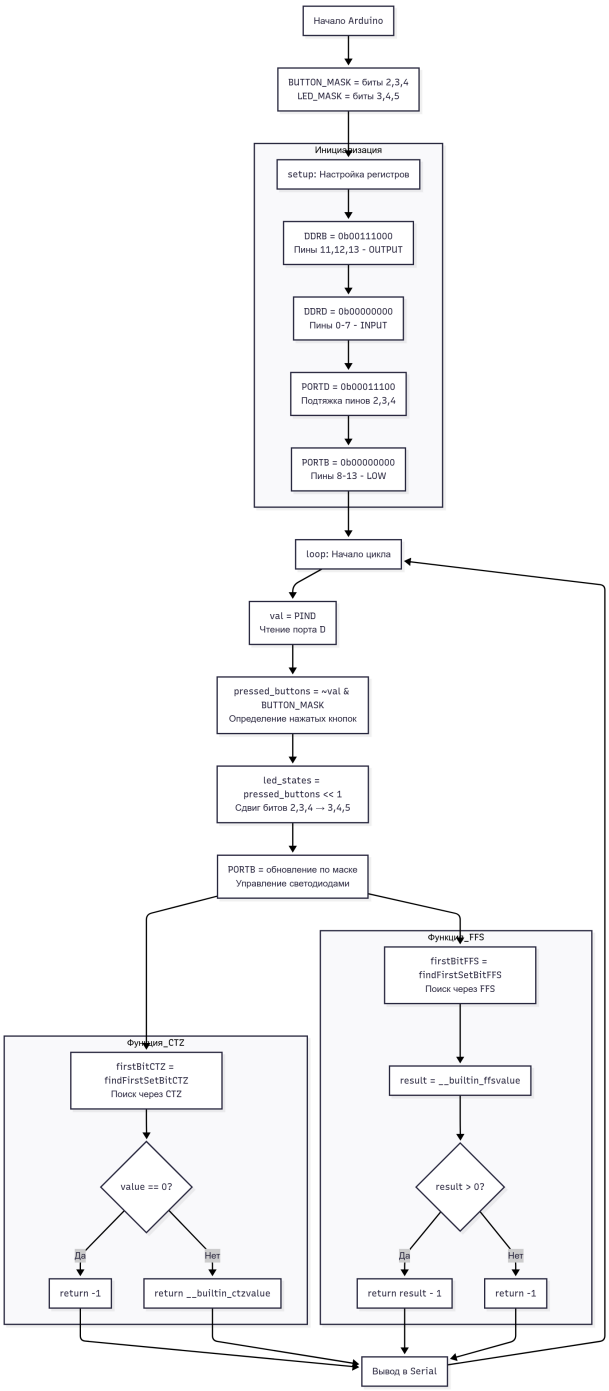

Начало Arduino
BUTTON_MASK = биты 2,3,4
LED_MASK = биты 3,4,5
Инициализация
setup: Настройка регистров
DDRB = 0b00111000
Пины 11,12,13 - OUTPUT
DDRD = 0b00000000
Пины 0-7 - INPUT
PORTD = 0b00011100
Подтяжка пинов 2,3,4
PORTB = 0b00000000
Пины 8-13 - LOW
loop: Начало цикла
val = PIND
Чтение порта D
pressed_buttons = ~val &
BUTTON_MASK
Определение нажатых кнопок
led_states =
pressed_buttons << 1
Сдвиг битов 2,3,4 → 3,4,5
PORTB = обновление по маске
Управление светодиодами
Функция_поиска_бита
firstBit =
findFirstSetBitLoop
Поиск первого
установленного бита
value == 0?
Нет
Цикл for i=0 до 7
Да
Бит i установлен?
Да
Нет
Да
i < 7?
Нет
return -1
return i
return -1
Вывод нажатых кнопок в
Serial
Вывод первого бита в Serial

```
Нажатые кнопки: 00010000 | Первый бит (цикл): D4
Нажатые кнопки: 00010000 | Первый бит (цикл): D4
Нажатые кнопки: 00011000 | Первый бит (цикл): D3
Нажатые кнопки: 00011000 | Первый бит (цикл): D3
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00010100 | Первый бит (цикл): D2
Нажатые кнопки: 00010100 | Первый бит (цикл): D2
Нажатые кнопки: 00010100 | Первый бит (цикл): D2
Нажатые кнопки: 00000100 | Первый бит (цикл): D2
Нажатые кнопки: 00000000 | Первый бит (цикл): нет
Нажатые кнопки: 00010000 | Первый бит (цикл): D4
Нажатые кнопки: 00011000 | Первый бит (цикл): D3
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00011100 | Первый бит (цикл): D2
Нажатые кнопки: 00000000 | Первый бит (цикл): нет
Нажатые кнопки: 00000000 | Первый бит (цикл): нет
```

Способ 2: через __builtin_ctz/__builtin_ffs

```
const uint8_t BUTTON_MASK = (1 << PD2) | (1 << PD3) | (1 << PD4);
const uint8_t LED_MASK = (1 << PB3) | (1 << PB4) | (1 << PB5);

int findFirstSetBitCTZ(uint8_t value) {
  if (value == 0) return -1;
  return __builtin_ctz(value);
}

int findFirstSetBitFFS(uint8_t value) {
  int result = __builtin_ffs(value);  // Индекс с 1
  return result > 0 ? result - 1 : -1;  // Индекс с 0
}

void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
  PORTB = 0b00000000;
  Serial.begin(9600);
}

void loop() {
  uint8_t val = PIND;
  uint8_t pressed_buttons = ~val & BUTTON_MASK;
  uint8_t led_states = pressed_buttons << 1;

  PORTB = (PORTB & ~LED_MASK) | (led_states & LED_MASK);

  int firstBitCTZ = findFirstSetBitCTZ(pressed_buttons);
  int firstBitFFS = findFirstSetBitFFS(pressed_buttons);

  Serial.print("Нажатые кнопки: ");
  for (int i = 7; i >= 0; i--) Serial.print((pressed_buttons >> i) & 1);
  Serial.print(" | CTZ: ");
  if (firstBitCTZ != -1) Serial.print("D" + String(firstBitCTZ));
  else Serial.print("нет");
  Serial.print(" | FFS: ");
```

```
  if (firstBitFFS != -1) Serial.print("D" + String(firstBitFFS));
  else Serial.print("нет");
  Serial.println();

  delay(500);
}
```



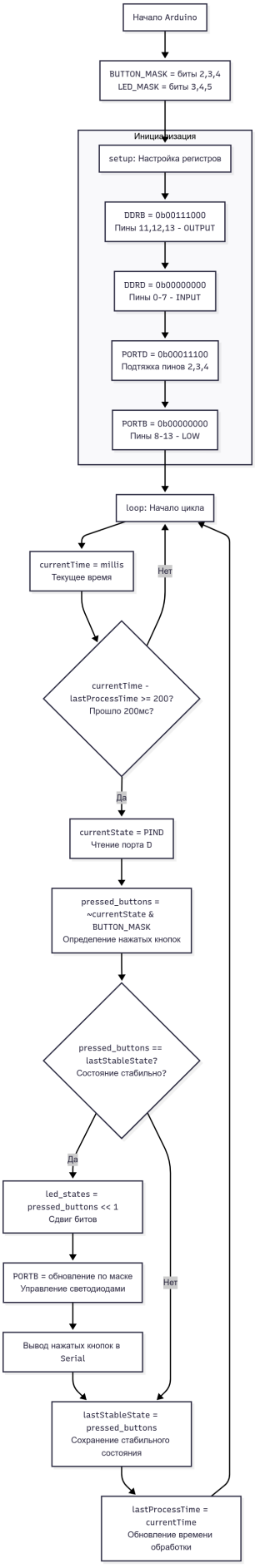
```
Нажатые кнопки: 00000000 | CTZ: нет | FFS: нет
Нажатые кнопки: 00000000 | CTZ: нет | FFS: нет
Нажатые кнопки: 00000000 | CTZ: нет | FFS: нет
Нажатые кнопки: 00010000 | CTZ: D4 | FFS: D4
Нажатые кнопки: 00010000 | CTZ: D4 | FFS: D4
Нажатые кнопки: 00010000 | CTZ: D4 | FFS: D4
Нажатые кнопки: 00011000 | CTZ: D3 | FFS: D3
Нажатые кнопки: 00011000 | CTZ: D3 | FFS: D3
Нажатые кнопки: 00011100 | CTZ: D2 | FFS: D2
Нажатые кнопки: 00001100 | CTZ: D2 | FFS: D2
Нажатые кнопки: 00011100 | CTZ: D2 | FFS: D2
Нажатые кнопки: 00011100 | CTZ: D2 | FFS: D2
Нажатые кнопки: 00000000 | CTZ: нет | FFS: нет
Нажатые кнопки: 00000000 | CTZ: нет | FFS: нет
Нажатые кнопки: 00000000 | CTZ: нет | FFS: нет
```

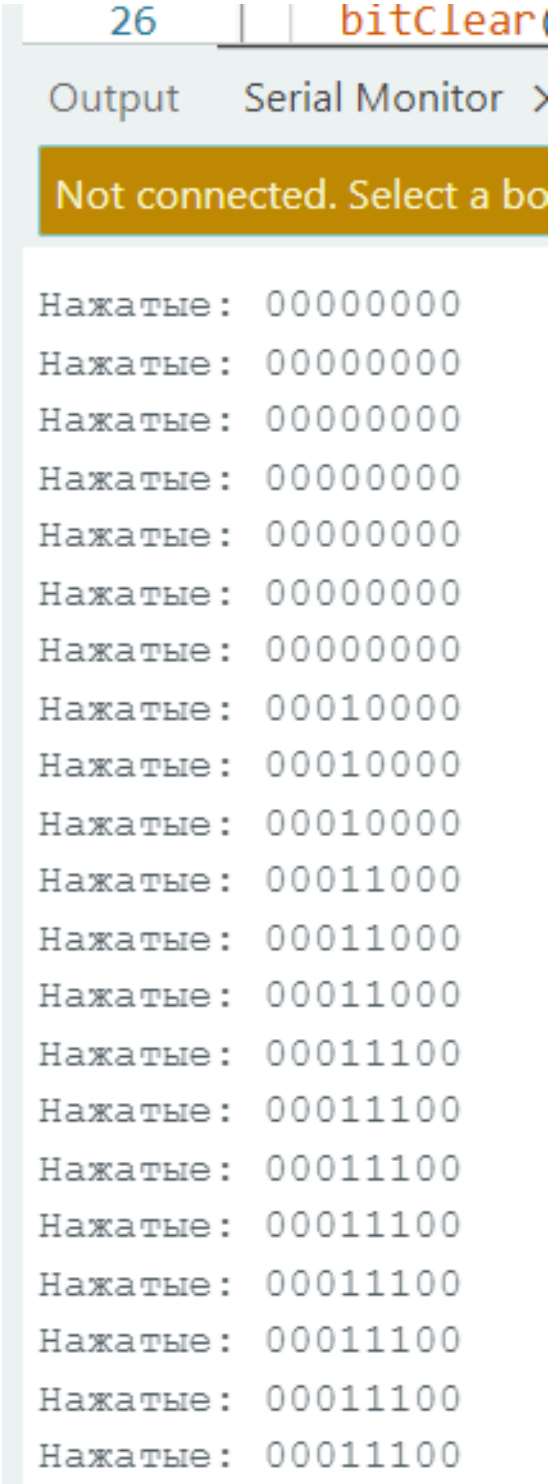
Задание 2.6:

Выполнять операцию циклично с частотой не менее 5 раз в секунду, при этом обеспечить устойчивое считывание (debounce) для кнопок программно.

```
const uint8_t BUTTON_MASK = (1 << PD2) | (1 << PD3) | (1 << PD4);
const uint8_t LED_MASK = (1 << PB3) | (1 << PB4) | (1 << PB5);

uint32_t lastProcessTime = 0;
uint8_t lastStableState = 0;

void setup() {
  DDRB = 0b00111000;
  DDRD = 0b00000000;
  PORTD = 0b00011100;
  PORTB = 0b00000000;
  Serial.begin(9600);
}

void loop() {
  uint32_t currentTime = millis();

  // Гарантируем частоту 5 Гц (каждые 200ms)
  if (currentTime - lastProcessTime >= 200) {

    uint8_t currentState = PIND;
    uint8_t pressed_buttons = ~currentState & BUTTON_MASK;

    // Простой антидребезг - обновляем только если состояние стабильно
    if (pressed_buttons == lastStableState) {
      // Состояние стабильно - обрабатываем
      uint8_t led_states = pressed_buttons << 1;
      PORTB = (PORTB & ~LED_MASK) | (led_states & LED_MASK);

      // Вывод в терминал
      Serial.print("Нажатые: ");
      for (int i = 7; i >= 0; i--) {
        Serial.print((pressed_buttons >> i) & 1);
      }
      Serial.println();
    }

```

```
38      lastStableState = pressed_buttons;
39      lastProcessTime = currentTime;
40    }
41  }
```


Начало Arduino
BUTTON_MASK = биты 2,3,4
LED_MASK = биты 3,4,5
Инициализация
setup: Настройка регистров
DDRB = 0b00111000
Пины 11,12,13 - OUTPUT
DDRD = 0b00000000
Пины 0-7 - INPUT
PORTD = 0b00011100
Подтяжка пинов 2,3,4
PORTB = 0b00000000
Пины 8-13 - LOW
loop: Начало цикла
currentTime = millis
Текущее время
Нет
currentTime -
lastProcessTime >= 200?
Прошло 200мс?
Да
currentState = PIND
Чтение порта D
pressed_buttons =
~currentState &
BUTTON_MASK
Определение нажатых кнопок
pressed_buttons ==
lastStableState?
Состояние стабильно?
Да
led_states =
pressed_buttons << 1
Сдвиг битов
PORTB = обновление по маске
Управление светодиодами
Нет
Вывод нажатых кнопок в
Serial
lastStableState =
pressed_buttons
Сохранение стабильного
состояния
lastProcessTime =
currentTime
Обновление времени
обработки



Вывод: В ходе лабораторной работы были успешно освоены методы низкоуровневой работы с регистрами портов микроконтроллера ATmega328P. Практика показала преимущество прямого управления портами через регистры ввода-вывода по сравнению с использованием стандартных функций Arduino - такой подход обеспечивает более высокую скорость выполнения операций. Были отработаны ключевые приемы: чтение состояния всего порта байтом, анализ отдельных битов с помощью битовых масок и операций сдвига, а также точная установка битов в нужных позициях порта. Полученные навыки позволяют создавать более эффективные программы для задач, требующих высокого быстродействия и оп-

тимизации ресурсов микроконтроллера.

Контрольные вопросы

1. **Регистры управления портами:**

```
DDRx  - направление (1=выход, 0=вход)
PORTx - данные/подтяжка (1=HIGH/подтяжка, 0=LOW)
PINx  - чтение состояния пинов
```

2. **Сильная/слабая единица и Hi-Z:**

```
Сильная: MOSFET полностью открыт (низкое R)
Слабая: ток ограничен резистором
Hi-Z: высокий импеданс (отключен)
Важно: предотвращение конфликтов, экономия энергии
```

3. **Чтение порта:**

```
uint8_t val = PINC;  // Читаем весь порт C
PINx возвращает текущее логическое состояние пинов
```

4. **Проверка бита:**

```
// Способ 1:
if (val & (1 << n)) { /* бит установлен */ }

// Способ 2:
if (val & _BV(n)) { /* бит установлен */ }
```

5. **Установка бита:**

```
PORTB |= (1 << n);  // Установить бит n в PORTB
```

6. **Отключение прерываний:**

```
// Необходимо для атомарности операций с портами
cli();  // Отключить прерывания
PORTD = new_value;
sei();  // Включить прерывания
```

7. **Плавающий вход:**

```
Риски: случайные срабатывания, повышенное потребление
Решение: включить подтяжку:
DDRB &= ~(1 << pin);   // Вход
PORTB |= (1 << pin);   // Включить подтяжку
```

8. **Debounce для кнопок:**

```
// Аппаратный: RC-фильтр
// Программный:
if (PINB & (1 << BUTTON_PIN)) {
    _delay_ms(50);  // Задержка для стабилизации
    if (PINB & (1 << BUTTON_PIN)) {
        // Обработка нажатия
    }
}
```

9. **PORTx/DDRx vs digitalWrite()/pinMode():**

```
Преимущества: быстрее, меньше кода, прямой контроль
Недостатки: менее читаемо, зависит от архитектуры
```

10. **Индекс первого установленного бита:**

```
uint8_t find_first_set(uint8_t byte) {
    if (byte == 0) return 255;
    uint8_t pos = 0;
    while ((byte & 1) == 0) {
        byte >>= 1;
        pos++;
    }
    return pos;
}
```