

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени  
И.С.ТУРГЕНЕВА»**

**ИНСТИТУТ ПРИБОРОСТРОЕНИЯ, АВТОМАТИЗАЦИИ И  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра программной инженерии

**Захарова О.В.**

**Методические указания к лабораторным работам  
по дисциплине «Программирование  
микроконтроллеров»**

*для направлений подготовки: 09.03.01, 09.03.04*

**Орел 2020**

## **Содержание**

### **Лабораторная работа № 1-2**

**Изучение документации к микроконтроллеру. Разработка программы на языке Си, работающей с портами ввода-вывода** **3**

**I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ** **3**

**II ЗАДАНИЕ** **22**

### **Лабораторная работа № 3**

**Разработка программы на языке Си, работающей с внешними прерываниями** **24**

**I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ** **24**

**II ЗАДАНИЕ** **29**

### **Лабораторная работа № 4**

**Разработка программы на языке Си, работающей с внешними прерываниями и таймерами** **30**

**I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ** **30**

**II ЗАДАНИЕ** **39**

### **Лабораторная работа № 5**

**Разработка программы на языке Си, работающей с приемопередатчиком (USART)** **40**

**I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ** **40**

**II ЗАДАНИЕ** **46**

# Лабораторная работа № 1-2

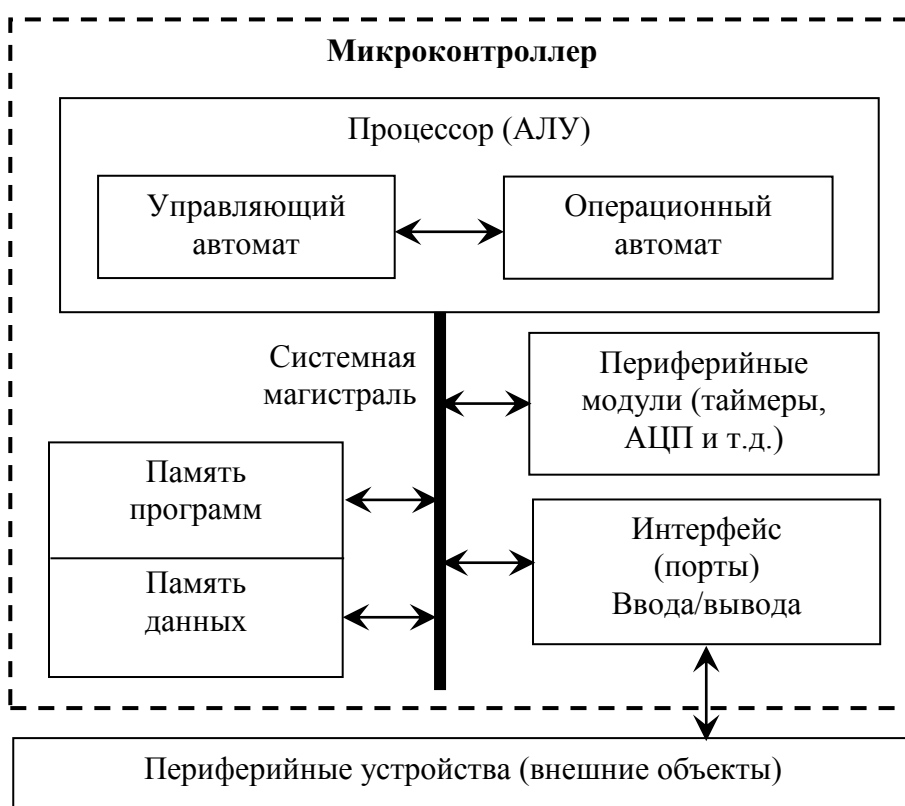
## Изучение документации к микроконтроллеру. Разработка программы на языке Си, работающей с портами ввода-вывода

### I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

#### 1 Структура микроконтроллера. Микроконтроллер ATtiny2313

##### 1.1 Обобщенная структура микроконтроллера

Структура микроконтроллера (МК) включает процессор (арифметико-логическое устройство – АЛУ, CPU), память программ и данных, периферийные модули, порты ввода/вывода (рис. 1.1).



**Рис 1.1.** Обобщенная схема микроконтроллера

Структура процессора (АЛУ) МК образуется управляющим и операционным автоматами. Управляющий автомат (управляющее устройство процессора) выполняет функции считывания и декодирования команд из памяти программ, а операционный автомат (операционное устройство процессора) отвечает за выполнение арифметических и логических операций. Системная магистраль состоит из шин адреса, данных и управления.

В памяти программ МК хранится управляющая программа, а память данных МК предназначена для чтения, записи и хранения данных, используемых программой.

Порты ввода/вывода МК необходимы для обеспечения сопряжения МК с периферийными устройствами (внешними объектами).

## 1.2 Обобщенная структура микроконтроллеров AVR

Обобщенная структура МК AVR включает (рис. 1.2):

- RISC-процессор. Идея RISC-архитектуры заключается в использовании упрощенного набора команд для минимизации времени выполнения вычислений (подбираются быстродействующие команды, выполняемые за один такт). RISC-процессор имеет упрощенную аппаратную реализацию, за счет чего достигается быстродействие вычислений, сокращается количество транзисторов, снижается потребляемая мощность и стоимость;
- Flash-память (FlashROM) – энергонезависимая память программ, выполненная по технологии Flash;
- память EEPROM – энергонезависимая память данных, выполненная по технологии EEPROM;
- память SRAM – оперативная память статического типа для хранения данных;
- интерфейс (порты) ввода/вывода. МК AVR имеют независимые линии, которые могут быть запрограммированы на вход или выход сигнала;
- выводы VCC и GND, предназначенные для подключения источника питания микроконтроллера.

МК AVR реализованы на основе Гарвардской архитектуры, то есть разделены: память программ (Flash-память) и память данных (SRAM и EEPROM); шины доступа к памяти – на шину управления и шину данных.

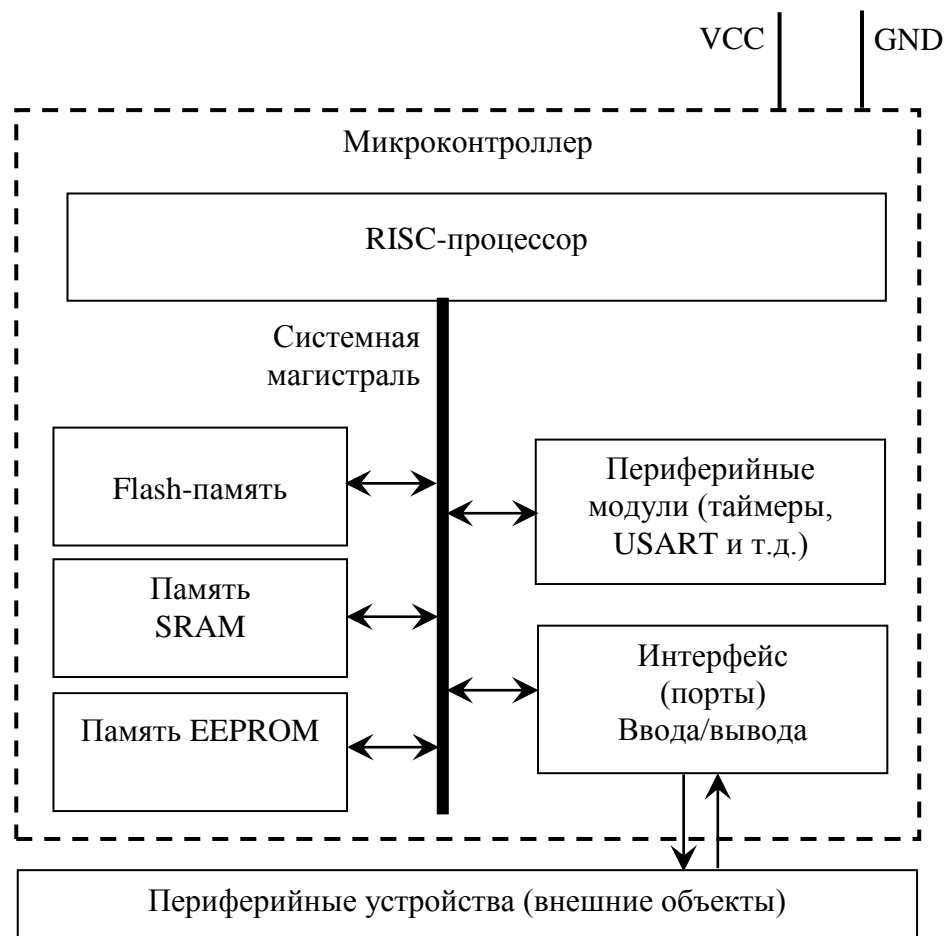
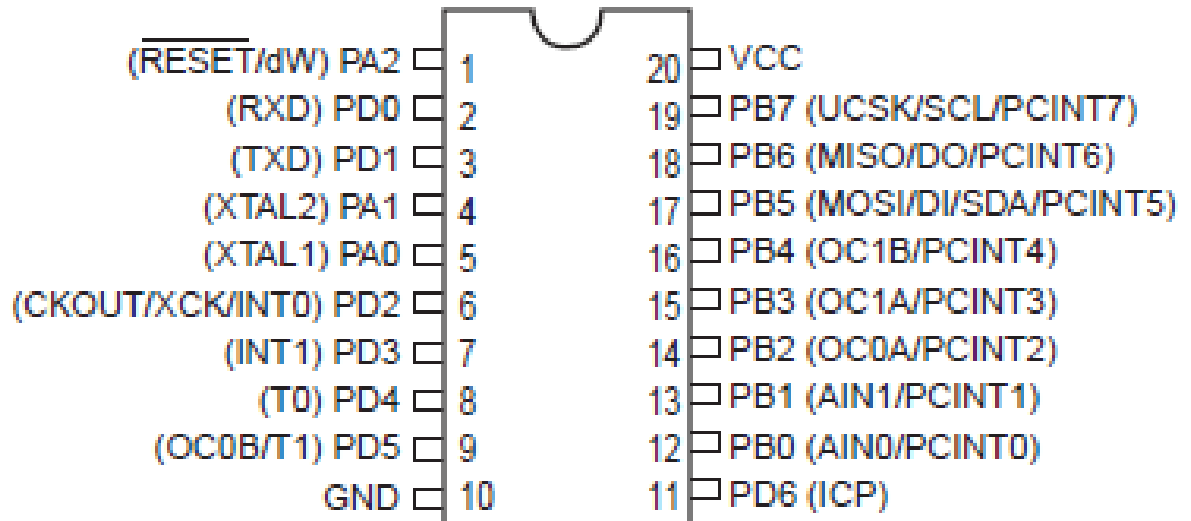


Рис. 1.2. Обобщенная структура микроконтроллеров AVR

## 1.3 Микроконтроллер ATtiny2313

### 1.3.1 Схема микроконтроллера ATtiny2313

Расположение выводов микроконтроллера ATtiny2313 в корпусе PDIP представлено на рис. 1.3.



Основные выводы: 1) VCC – подвод питающего напряжения;  
2) GND – заземление; 3) Port A (PA0 – PA2) – 3-битный двунаправленный порт ввода/вывода; 4) Port B (PB0 – PB7) – 8-битный двунаправленный порт ввода/вывода; 5) Port D (PD0 – PD6) – 7-битный двунаправленный порт ввода/вывода

**Рис. 1.3.** Расположение выводов микроконтроллера ATtiny2313 в корпусе PDIP

### 1.3.2 Порты ввода – вывода

Порты микроконтроллеров AVR представляют собой двунаправленные линии, которые могут работать как в режиме ввода, так и в режиме вывода, то есть каждый разряд порта можно настроить на вход или на выход.

Для каждого порта микроконтроллера имеется три регистра:

- 1) PORTx – регистр данных;
- 2) DDRx – регистр управления;
- 3) PINx – регистр чтения состояния линий порта.

Настройка вывода микроконтроллера на вывод информации осуществляется установкой бит регистра DDRx, а настройка на ввод – сбросом бит регистра DDRx.

Регистры портов ввода/вывода для микроконтроллера ATtiny2313 представлены на рис. 1.4 – 1.6.

Пример настройки порта B на вывод данных на языке Си:

$$DDRB = 0xFF;$$

Пример настройки четырех бит порта B на вывод данных (выводы PB0 - PB3) и четырех бит на считывание информации (выводы PB4 - PB7) на языке Си:

$$DDRB = 0x0F;$$

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	DDA2	DDA1	DDA0	DDRA
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

*Рис. 1.4.* Регистры порта А

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

*Рис. 1.5.* Регистры порта В

Bit	7	6	5	4	3	2	1	0	
	–	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	–	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	–	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

*Рис. 1.6.* Регистры порта D

## 2 Особенности языка Си для программирования микроконтроллеров

### 2.1. Структура программы

Программа на языке Си имеет структуру, представленную на рис. 2.1.

```
//директивы препроцессора
#директива_1
...
#директива_N

...
//объявление глобальных типов, констант и переменных
...

//функции
тип_функции_1 имя_функции_1 (формальные параметры)
{тело_функции}
...
тип_функции_N имя_функции_N (формальные параметры)
{тело_функции}

//главная функция
//с которой начинается выполнение программы
int main(void)
{
    //объявление локальных типов, констант и переменных
    ...
    while(1) //бесконечный цикл
    {
        ...
        //операторы
        ...
    }
}
```

**Рис 2.1.** Структура программы на языке Си

### 2.2. Системы счисления

Для представления числа в двоичной системе счисления используется префикс «0b», а для шестнадцатеричного представления – «0x». Примеры записи чисел в различных форматах представлены в табл. 2.1.

Десятичное, двоичное или шестнадцатеричное представление чисел при написании кода программы никакой вычислительной нагрузки не несет. Различие заключается в формате записи и удобстве восприятия представления чисел.

Таблица 2.1

*Представление чисел в различных системах счисления*

Десятичное представление	Двоичное представление	Шестнадцатеричное представление
0	0b00000000	0x00
1	0b00000001	0x01
2	0b00000010	0x02
3	0b00000011	0x03
4	0b00000100	0x04
5	0b00000101	0x05
6	0b00000110	0x06
7	0b00000111	0x07
8	0b00001000	0x08
9	0b00001001	0x09
10	0b00001010	0x0A
11	0b00001011	0x0B
12	0b00001100	0x0C
13	0b00001101	0x0D
14	0b00001110	0x0E
15	0b00001111	0x0F
...	...	...
253	0b11111101	0xFD
254	0b11111110	0xFE
255	0b11111111	0xFF

**2.3 Работа с разрядами для программирования микроконтроллеров****2.3.1 Оператор поразрядного сдвига влево**

Бинарный оператор  $\ll$  служит для поразрядного сдвига влево левого операнда на количество разрядов, указанное в правом операнде. Освободившиеся справа разряды в результате заполняются логическими «0».

Например, поразрядный сдвиг влево значения переменной  $X$  на две позиции ( $X \ll 2$ ) означает перестановку значения каждого разряда переменной  $X$  влево на две позиции и заполнение логическими «0» освободившихся левых бит.

Пусть  $X = 5_{10}$  (в десятичном представлении), тогда для вычисления  $X \ll 2$  необходимо выполнить следующие действия:

1) представить число, записанное в переменную  $X$ , в двоичной системе счисления (рис. 2.2):

$$5_{10} = 101_2;$$

2) выполнить сдвиг двоичного представления числа, записанного в переменную  $X$ , на две позиции влево и заполнить логическими «0» освободившиеся левые биты:

$$101_2 \ll 2_{10} = 10100_2;$$

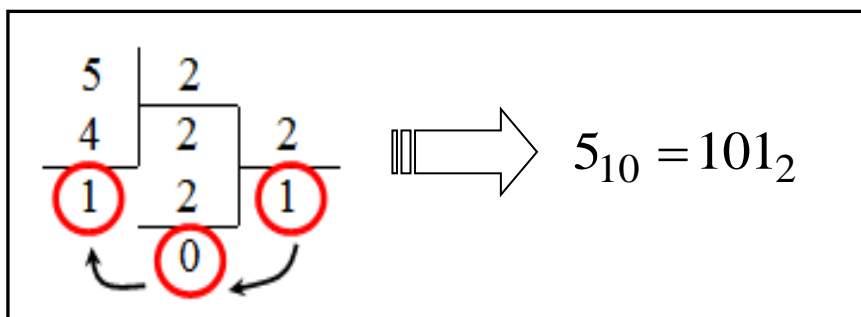
3) представить получившийся результат в десятичном представлении:



$$10100_2 = (1 \cdot 2)^4 + (0 \cdot 2)^3 + (1 \cdot 2)^2 + (0 \cdot 2)^1 + (0 \cdot 2)^0 = \\ = 16_{10} + 0_{10} + 4_{10} + 0_{10} + 0_{10} = 20_{10}.$$

То есть:

$$5 \ll 2 = 20.$$



**Рис. 2.2.** Перевод числа 5 в двоичную систему счисления

Следует заметить, что сдвиг влево десятичного значения на две позиции эквивалентен умножению значения первого операнда на  $2^2$ , то есть

$$X \ll 2 = X \cdot 2^2 = X \cdot 4,$$

а сдвиг влево на три позиции – умножению на  $2^3$ , то есть:

$$X \ll 3 = X \cdot 2^3 = X \cdot 8.$$

### **Пример 1:**

```
char n = 0b00001011;
n = (n << 3);           // n=0b01101000
```

### **Пример 2:**

```
char b = 0x06; // 0x06 = 0b00000110
b = (b << 6);  // b=0b10000000
```

## **2.3.2 Оператор поразрядного сдвига вправо**

Бинарный оператор  $\gg$  служит для поразрядного сдвига вправо левого операнда на количество разрядов, указанное в правом операнде. Освободившиеся слева разряды заполняются логическими «0».

Например, поразрядный сдвиг вправо значения переменной  $X$  на две позиции ( $X \gg 2$ ) означает перестановку значения каждого разряда переменной  $X$  вправо на две позиции и заполнение логическими нулями освободившихся правых бит.

Пусть  $X = 5_{10}$ , тогда для вычисления  $X \gg 2$  необходимо выполнить следующие действия:

1) представить число, записанное в переменную  $X$ , в двоичной системе счисления (см. рис. 2.2):

$$5_{10} = 101_2;$$

2) выполнить сдвиг двоичного представления числа, записанного в переменную  $X$ , на две позиции вправо и заполнить логическими «0» освободившиеся правые биты:

$$101_2 \gg 2_{10} = 001_2;$$

3) представить получившийся результат в десятичном представлении:

$$5 \gg 2 = 1.$$

Следует заметить, что сдвиг вправо десятичного значения на две позиции эквивалентен целочисленному делению значения первого операнда на  $2^n$ .

**Пример 1:**

```
char n = 0b00001011;
n = (n >> 2);          // n=0b00000010
```

**Пример 2:**

```
char b = 0xFF; // 0x06 = 0b11111111
n = (n >> 6);   // n=0b00000011
```

### 2.3.3 Оператор побитового отрицания

Унарный оператор  $\sim$  служит для инвертирования каждого разряда операнда (заменяет логический «0» на логическую «1», а логическую «1» – на логический «0»).

Например:

```
n = 0b11001100;
n = ~n;          // n=0b00110011
```

### 2.3.4 Оператор поразрядной дизъюнкции

Бинарный оператор  $|$  выполняет поразрядную дизъюнкцию (поразрядное «ИЛИ») операндов. У результата соответствующий разряд принимает значение логической «1», если логическая «1» содержится в соответствующем разряде хотя бы одного исходного операнда, в противном случае соответствующий разряд результата равен логическому «0».

Например:

```
A = 0b00010011;
B = 0b00100001;
C = A | B;       // C=0b00110011
```

### 2.3.5 Оператор поразрядной конъюнкции

Бинарный оператор  $\&$  выполняет поразрядную конъюнкцию (поразрядное «И») операндов. У результата соответствующий разряд принимает значение логического «0», если логический «0» содержится в соответствующем разряде хотя бы одного исходного операнда, в противном случае соответствующий разряд результата равен логической «1».

Например:

```
A = 0b00010011;  
B = 0b00100001;  
C = A & B;      // C=0b00000001
```

### 2.3.6 Оператор поразрядного исключающего «ИЛИ»

Бинарный оператор  $\wedge$  выполняет поразрядную логическую операцию исключающего «ИЛИ» над исходными операндами. У результата соответствующий разряд принимает значение логической «1», если соответствующие разряды исходных операндов различаются, и логического «0» – в противном случае (то есть если соответствующие разряды у операндов принимают одинаковые значения).

Например:

```
A = 0b00010011;  
B = 0b00100001;  
C = A ^ B;      // C=0b00110010
```

### 2.3.7 Установка разрядов

Установка определенного разряда (записи логической «1» в определенный разряд) с обнулением остальных бит:

$$\text{результат} = (1 \ll \text{номер\_разряда});$$

Например, установка пятого разряда с обнулением остальных бит:

```
n = (1 << 5); // n = 0n00100000
```

**Установка соответствующих разрядов с обнулением остальных разрядов:**

Вариант 1:

$$\text{результат} = \text{требуемое\_значение};$$

Например, установка первого, третьего и пятого бит:

```
n = 0n00101010;
```

Вариант 2:

$$\text{результат} = (1 \ll \text{номер\_разряда}) | \dots | (1 \ll \text{номер\_разряда});$$

Например, установка первого, третьего и пятого бит:

```

n = (1<<1) / (1<<3) / (1<<5); // 1. (1<<1) = 0b00000010;
                                // 2. (1<<3) = 0b00001000;
                                // 3. (1<<5) = 0b00100000;
                                // 4. n = (0b00000010) / (0b00001000) / (0b00100000) =
                                // = 0b00101010 – результат.

```

**Установка определенного разряда без обнуления остальных:**

Вариант 1:

```
результат = операнд | (1<<номер_разряда);
```

Вариант 2:

```
результат |= (1<<номер_разряда),
```

Например, установка второго бита без обнуления остальных разрядов:

```

n = 0b00110001;
n /= (1 << 2);    // n = 0b00110101

```

**Установка нескольких бит без обнуления остальных разрядов:**

Вариант 1:

```
результат = операнд | (1<<номер_разряда) | ... | (1<<номер_разряда);
```

Вариант 2:

```
результат |= (1<<номер_разряда) | ... | (1<<номер_разряда);
```

Например, установка второго и третьего бита без обнуления остальных разрядов:

```

n = 0b00100001;
n /= (1 << 2) / (1 << 3); // n = 0b00101101

```

### 2.3.8 Сброс разрядов

**Сброс определенного разряда без обнуления остальных разрядов:**

Вариант 1:

```
результат = операнд & ~ (1<<номер_разряда);
```

Вариант 2:

```
результат &= ~ (1<<номер_разряда);
```

Например, сброс второго бита:

```

n = 0b00001111;
n &= ~(1 << 2); // n = 0b00001111 & 0b11111011 = 0b00001011

```



```

// настройка выводов порта B на вывод данных
DDRB = 0xFF;

...
// если третий разряд регистра PIND равен логическому «0»
// (кнопка не нажата), то включаем светодиоды
if ((PIND & (1 << 3)) == 0)
{PORTB = 0xFF;}
// в противном случае выключаем светодиоды
else
{PORTB = 0x00;}

```

Вариант 2: использование функции

*bit\_is\_clear(переменная номер\_разряда)*

из библиотеки `avr/sfr_defs.h`:

```

// если проверяемый разряд номер_разряда равен
// логическому «0», то выполнится действие
if (bit_is_clear(переменная, номер_разряда)) == 0)
{действие;}

```

или

```

// если проверяемый разряд номер_разряда равен
// логическому «0», то выполнится действие_1,
/// в противном случае будет выполнено действие_2
if (bit_is_clear(переменная, номер_разряда)) == 0)
{действие_1;}
else
{действие_2;}

```

Например:

```

#include <avr/sfr_defs.h>

...
DDRB = 0xFF;

...
if (bit_is_clear(PIND, 3))
{PORTB = 0xFF;}
else
{PORTB = 0x00;}

```

### 2.3.10 Проверка разряда на наличие логической «1»

Проверка разряда переменной на наличие логической «1» :

Вариант 1:

```
// если проверяемый разряд номер_разряда равен
// логической «1», то выполнится действие
if ((переменная & (1 << номер_разряда)) != 0)
    {действие;}
```

или

```
// если проверяемый разряд номер_разряда равен
// логической «1», то выполнится действие 1,
// в противном случае будет выполнено действие 2.
if ((переменная & (1 << номер_разряда)) != 0)
    {действие1;}
else
    {действие2;}
```

Например, вывод *PD3* подключен к кнопке, а выводы порта *B* – к светодиодам:

```
// настройка выводов порта B на вывод данных
DDRB = 0xFF;
...
// если третий разряд регистра PIND равен логической «1»
// (кнопка нажата), то включаем светодиоды
if ((PIND & (1 << 3)) != 0)
    {PORTB = 0xFF;}
// в противном случае выключаем светодиоды
else
    {PORTB = 0x00;}
```

Вариант 2: использование функции

*bit\_is\_set(переменная номер\_разряда)*

из библиотеки `avr/sfr_defs.h`:

```
// если проверяемый разряд номер_разряда равен
// логической «1», то выполнится действие
if (bit_is_set(переменная, номер_разряда))
    {действие;}
```

или

```
// если проверяемый разряд номер_разряда равен
// логической «1», то выполнится действие 1,
// в противном случае будет выполнено действие 2.
if (bit_is_set(переменная, номер_разряда))
    {действие1;}
```

```
else
{действие2;}
```

Например, вывод *PD3* подключен к кнопке, а выходы порта *B* – к светодиодам:

```
#include <avr/sfr_defs.h>
...
DDRB = 0xFF;
...
if (bit_is_set(PIND, 3))
{PORTB = 0xFF;}
else
{PORTB = 0x00;}
```

### 2.3.11 Ожидание появления логического «0» в разряде

Ожидание появления логического «0» в разряде переменной:

Вариант 1:

```
while (переменная & (1 << номер_разряда));
```

Цикл будет выполняться до тех пор, пока не появится логический «0» в разряде *номер\_разряда* переменной.

Вариант 2: использование функции из библиотеки *avr/sfr\_defs.h*:

```
loop_until_bit_is_clear(переменная, номер_разряда);
```

Например:

```
#include <avr/sfr_defs.h>
...
// Цикл будет выполняться до тех пор, пока во втором
// разряде регистра PIND не появится логический «0»
loop_until_bit_is_clear(PIND, 2);
```

### 2.3.12 Ожидание появления логической «1» в разряде

Ожидание появления логической «1» в разряде переменной:

Вариант 1:

```
while (~переменная & (1 << номер_разряда));
```

Цикл будет выполняться до тех пор, пока не появится логическая «1» в разряде *номер\_разряда* переменной.

Вариант 2: использование функции из библиотеки *avr/sfr\_defs.h*:

```
loop_until_bit_is_set(переменная, номер_разряда);
```

Например:



```
#include <avr/sfr_defs.h>

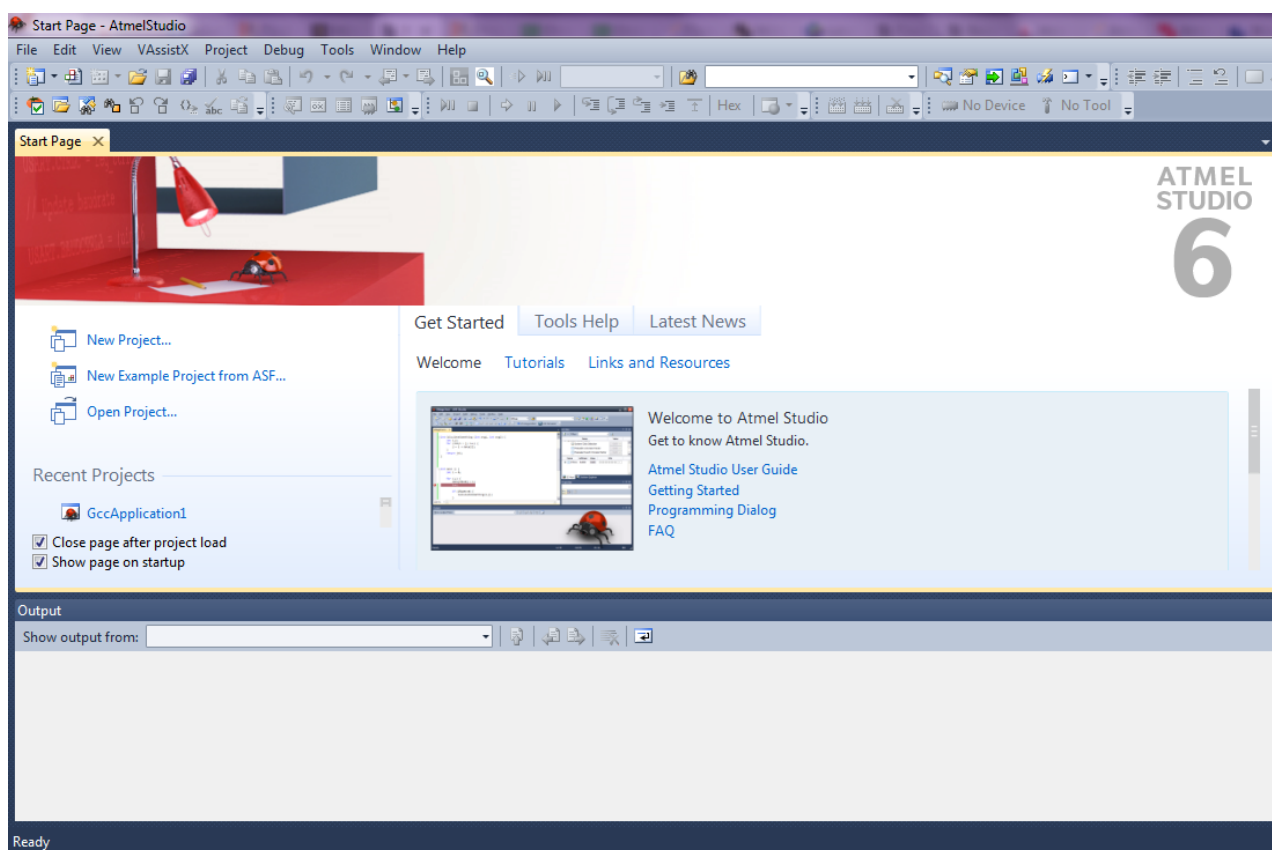
...
// Цикл будет выполняться до тех пор, пока во втором
// разряде регистра PIND не появится логическая «1»
loop_until_bit_is_set(n, 2);
```

### 3 Программное обеспечение ATMEL STUDIO

Atmel Studio – это бесплатное программное обеспечение для разработки программ для МК AVR и ARM на языках Си/Си++ или Ассемблера. В настоящей лабораторной работе рассматривается программное обеспечение Atmel Studio 6.0.

#### 3.1. Создание проекта

Стартовое окно программы Atmel Studio 6.0 представлено на рис. 3.1.



**Рис. 3.1.** Стартовое окно программы Atmel Studio 6.0

Для создания нового проекта необходимо выбрать пункт меню «Файл -> New -> Project...» или значок «New Project», после чего откроется окно мастера создания проектов (рис. 3.2). Далее выбирается язык программирования, тип программы, указывается название проекта, путь к папке и нажимается кнопка «ОК». В открывшемся окне необходимо выбрать название МК, для которого будет разрабатываться программа, и нажать кнопку «ОК» (рис. 3.3). Окно разработки программы представлено на рис. 3.4.

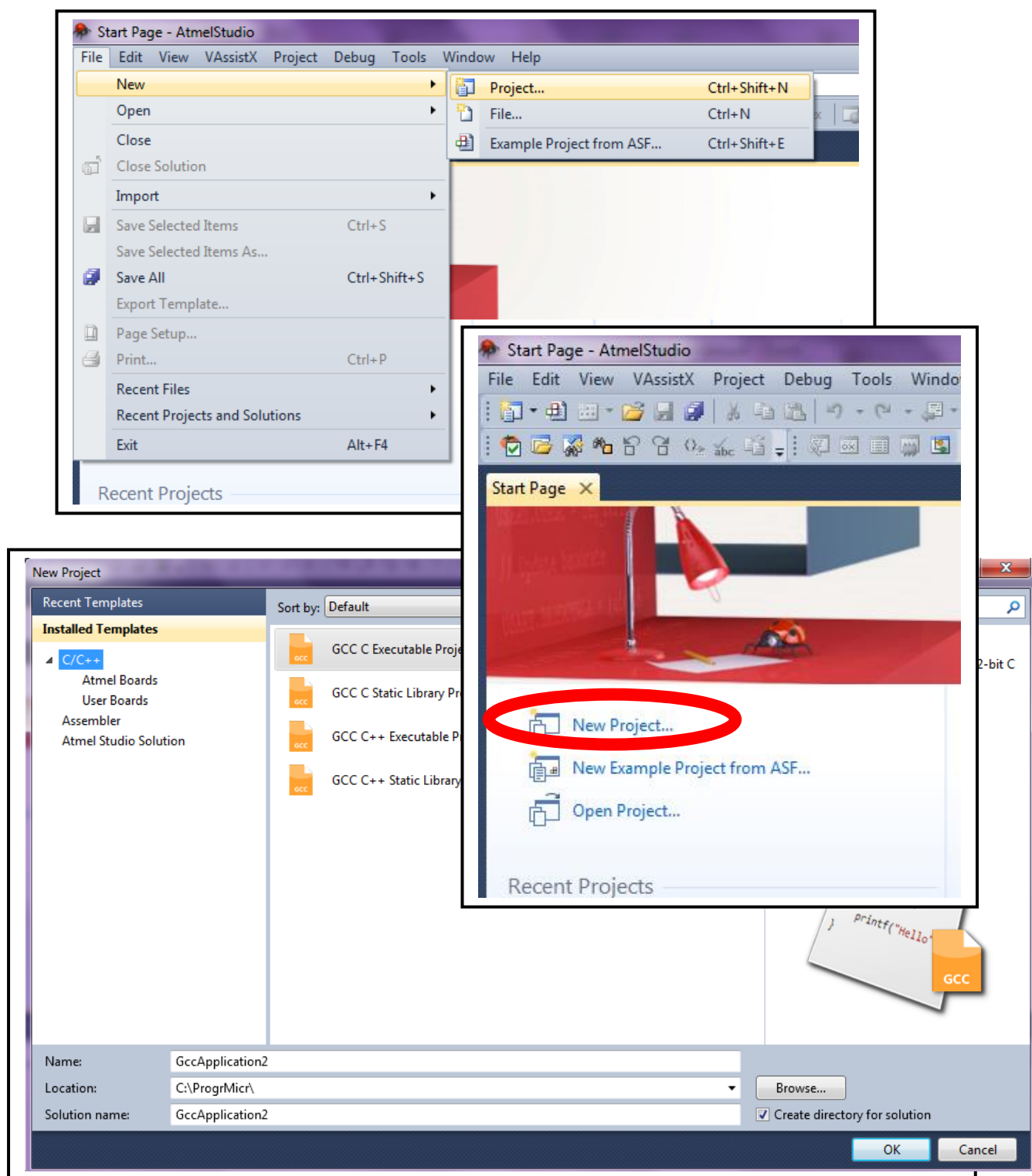


Рис. 3.2. Создание нового проекта

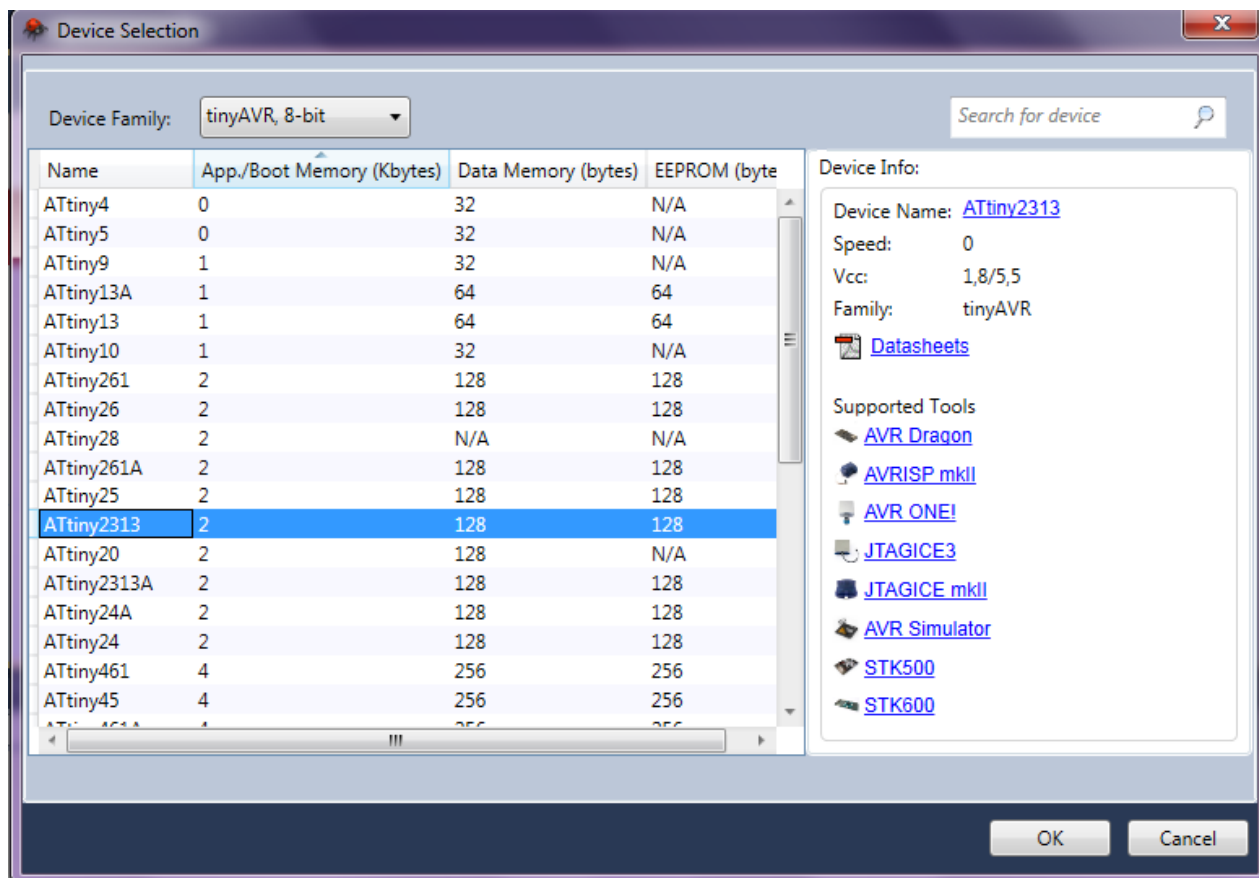


Рис. 3.3. Выбор микроконтроллера

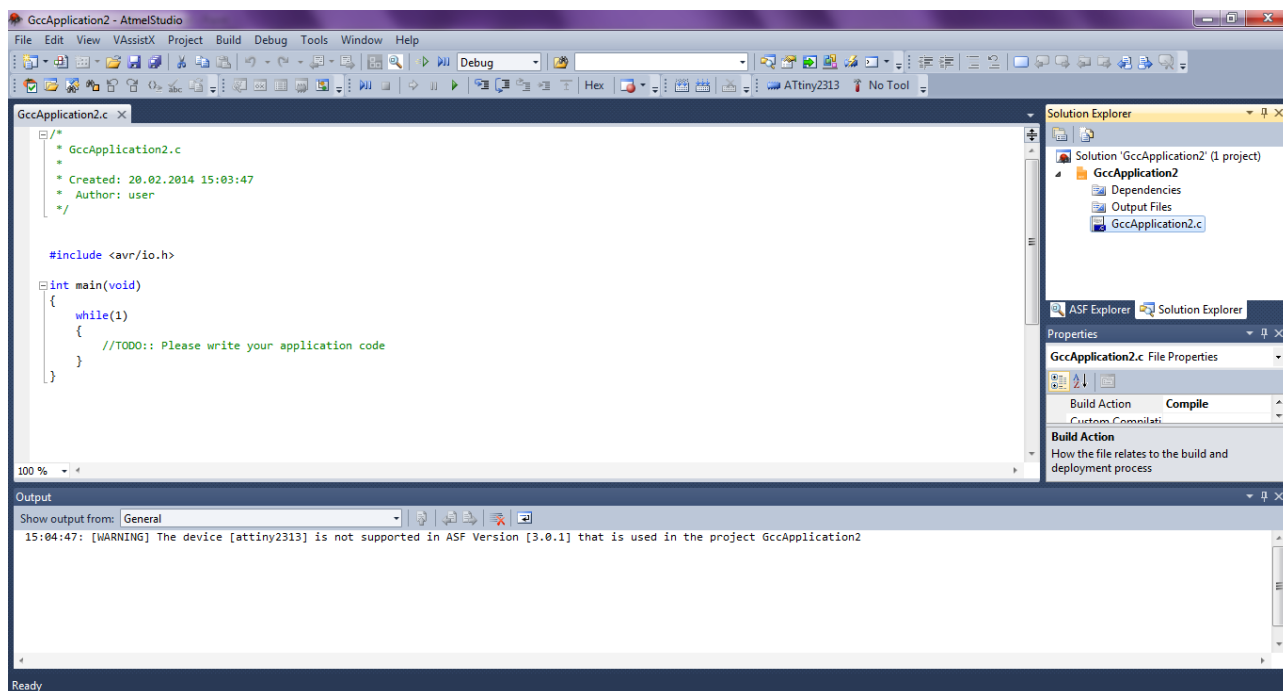
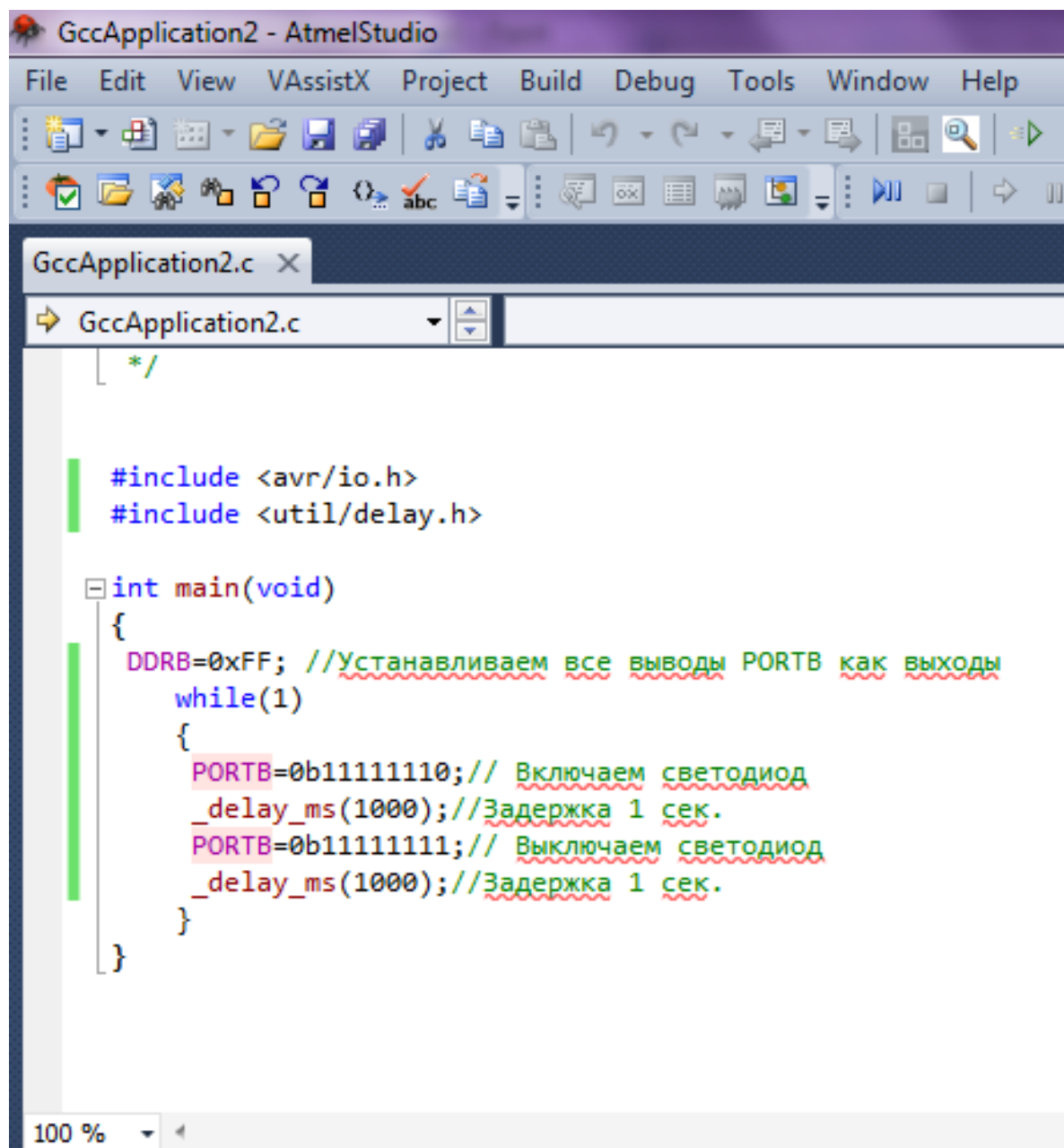


Рис. 3.4. Окно создания программ

### 3.2. Разработка программы и формирование hex-файла

После создания программного кода (например, рис. 3.5) необходимо выбрать пункт меню «Build → Build Solution», и если компилятор не обнаружит ошибок, то сформируется hex-файл, который будет прошиваться в МК (рис. 3.6).



```
GccApplication2 - AtmelStudio
File Edit View VAssistX Project Build Debug Tools Window Help
GccApplication2.c x
GccApplication2.c
*/

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

int main(void)
{
    DDRB=0xFF; //Устанавливаем все выходы PORTB как выходы
    while(1)
    {
        PORTB=0b11111110; // Включаем светодиод
        _delay_ms(1000); //Задержка 1 сек.
        PORTB=0b11111111; // Выключаем светодиод
        _delay_ms(1000); //Задержка 1 сек.
    }
}
```

Рис. 3.5. Пример программного кода

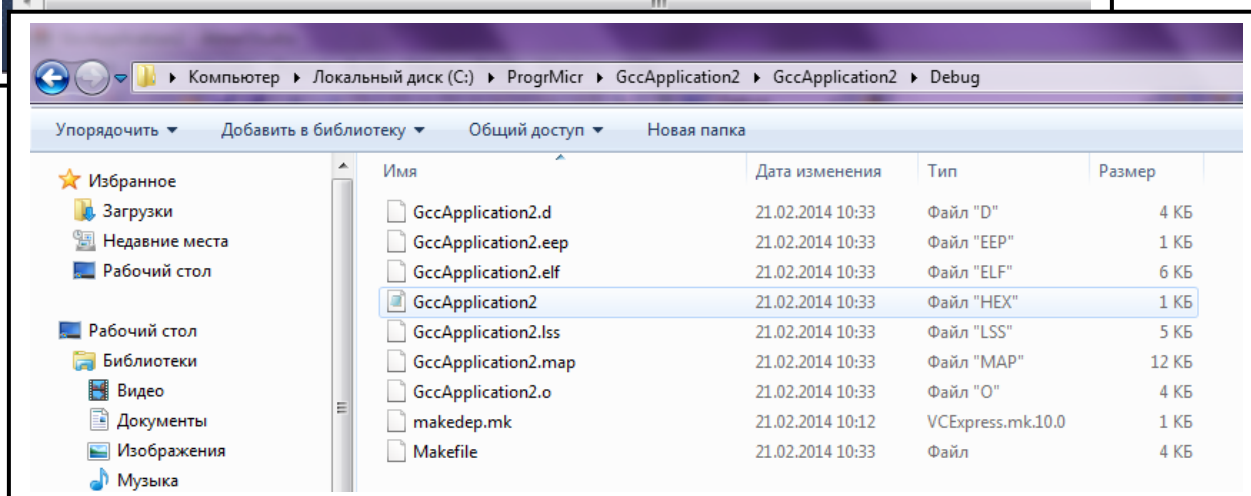
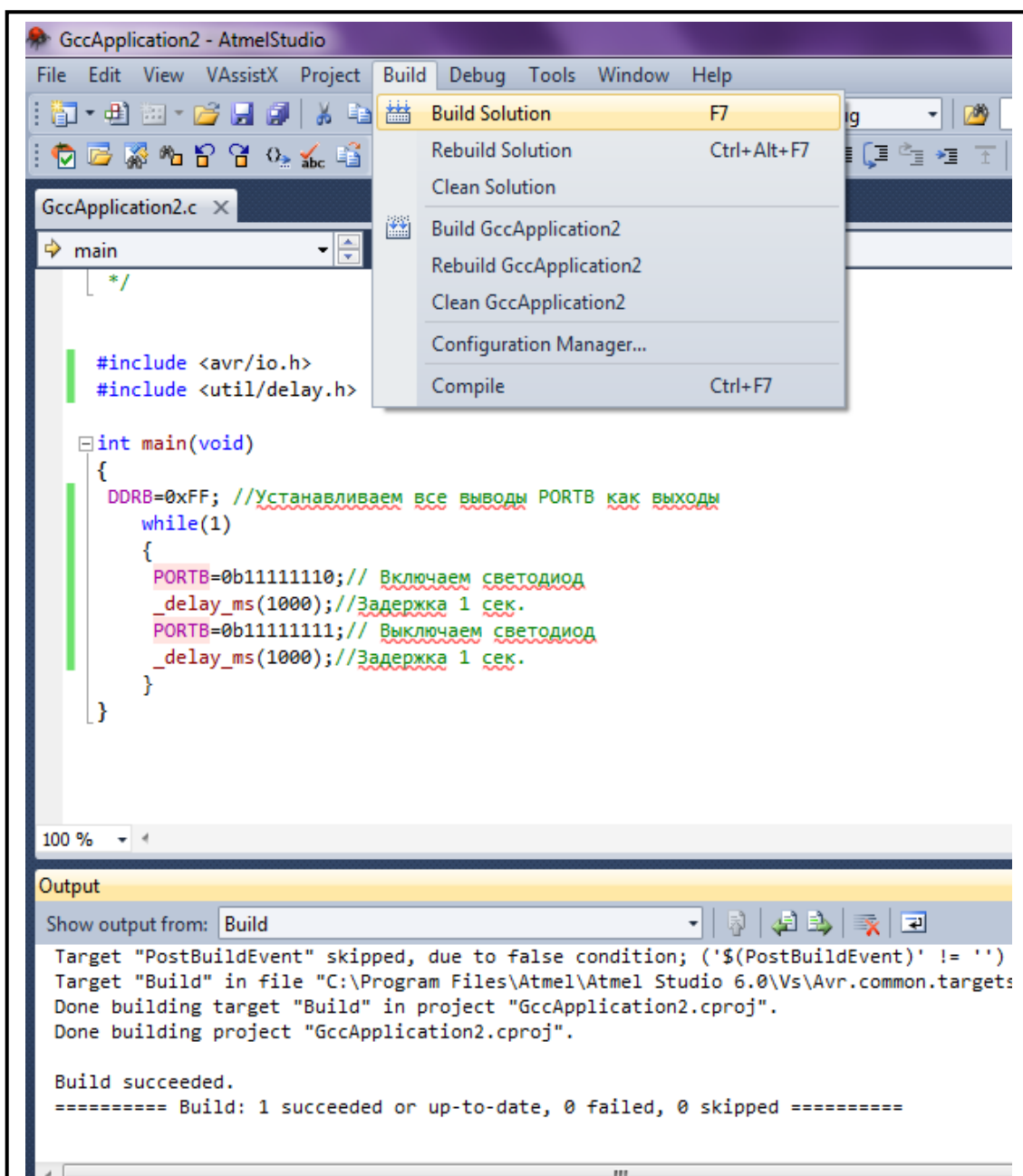
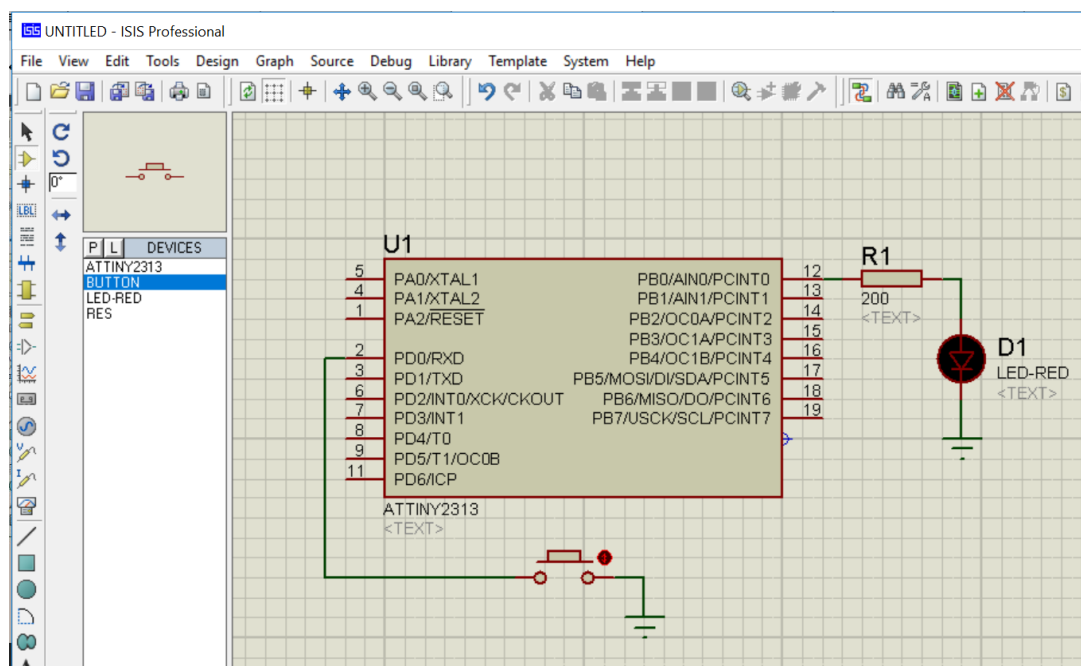


Рис. 3.6. Формирование hex -файла

#### 4 Система автоматизированного проектирования электронных схем Proteus

На рис. 4.1 представлен пример подключения светодиода и кнопки к микроконтроллеру ATtiny2313 в программе Proteus.



**Рис. 4.1.** Пример подключения светодиода и кнопки к МК ATtiny2313 в программе Proteus

Для прошивки программы в микроконтроллер необходимо: двойным кликом по микроконтроллеру вызвать окно «Edit Component», в поле «Program File» указать путь к hex-файлу; нажать кнопку «ОК» (рис. 4.2).

## II ЗАДАНИЕ

3.1. Изучить документацию к микроконтроллеру из таблицы 3.1. Номер варианта для выполнения лабораторных работ студент выбирает из Таблицы 3.1 в соответствии с номером в журнале.

Таблица 3.1

Варианты заданий

Вариант	Микроконтроллер	Вариант	Микроконтроллер
1	ATmega1281	12	ATmega1284P
2	ATmega6450	13	ATmega644
3	ATmega32	14	ATmega8515
4	ATmega103	15	ATmega48
5	ATmega329	16	ATmega2561
6	ATmega2560	17	ATmega3250`
7	ATmega645	18	ATmega644P
8	ATmega1280	19	ATmega16
9	ATmega6490	20	ATmega3290
10	ATmega329P	21	ATmega165P
11	ATmega169P	22	ATmega168

3.2. Изучить среду разработки программ для микроконтроллеров Atmel Studio. Разработать программу, позволяющую включать/выключать светодиод по нажатию на кнопку.

3.3. Изучить систему автоматизированного проектирования электронных схем Proteus. Разработать схему для проверки работоспособности программы из задания 3.2.

3.4. Для изученного микроконтроллера (Таблица 3.1) разработать программу на языке Си, позволяющую:

- включать/выключать светодиодную гирлянду при помощи Кнопки1;
- изменять режимы работы гирлянды при помощи Кнопки2;
- изменять скорость работы режимов гирлянды при помощи Кнопки3 (увеличивать/уменьшать скорость);

Придумать не менее трех режимов работы гирлянды (среди режимов должна быть светодиодная дорожка).

При разработке программы не использовать таймеры и внешние прерывания.

В САПР Proteus разработать схему для проверки работоспособности программы для микроконтроллера.

# Лабораторная работа № 3

## Разработка программы на языке Си, работающей с внешними прерываниями

### I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

#### Внешние прерывания МК ATtiny2313

Принято считать, что прерывание (interrupt) – это сигнал процессору о наступлении какого-либо события, причем выполнение текущей последовательности команд приостанавливается и управление передаётся обработчику прерывания, который реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код (рис. 1).

У МК ATtiny2313 внешние прерывания (ВП) делятся на настраиваемые (задействованы выходы *INT0* и *INT1*) и ненастраиваемые (задействованы выходы *PCINT0* – *PCINT7*) (рис. 2).

В регистре статуса *SREG* для глобального разрешения прерываний седьмой бит *I* должен быть установлен, а для глобального запрета прерываний – сброшен (рис. 3). Программная установка бита *I* реализуется командой *sei*, а сброс – командой *clic*.

Для настройки ВП *INT0* и *INT1* используется регистр управления ВП микроконтроллера *MCUCR* (рис. 4):

- Биты 1 и 0 (*ISC00*, *ISC01*) – настройка ВП *INT0*. Режимы настройки ВП *INT0* представлены в Табл. 1. Элементы цифрового сигнала представлены на рис. 5. ВП *INT0* будет вызвано на внешнем выводе *INT0*, если бит *INT0* в регистре маски ВП *GIMSK* и бит *I* в регистре статуса *SREG* будут установлены.

- Биты 3 и 2 (*ISC10*, *ISC11*) – настройка ВП *INT1*. Режимы настройки ВП *INT1* представлены в Табл. 2. ВП *INT1* будет вызвано на внешнем выводе *INT1*, если будут установлены бит *INT1* в регистре маски ВП прерываний *GIMSK* и бит *I* в регистре статуса *SREG*.

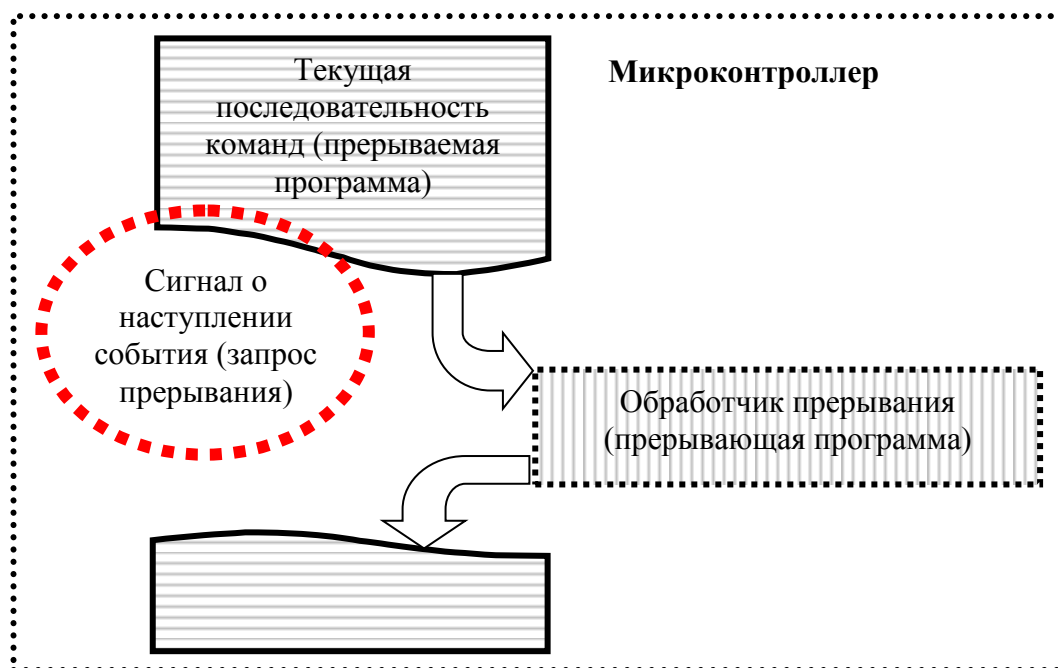


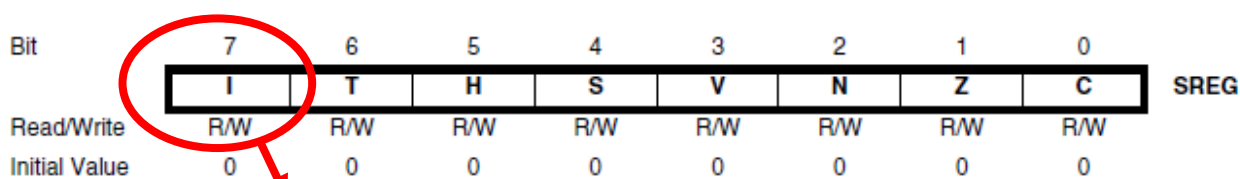
Рис. 1. Выполнение прерывания





Ненастраиваемое прерывание на выводах *PCINT0 – PCINT7*

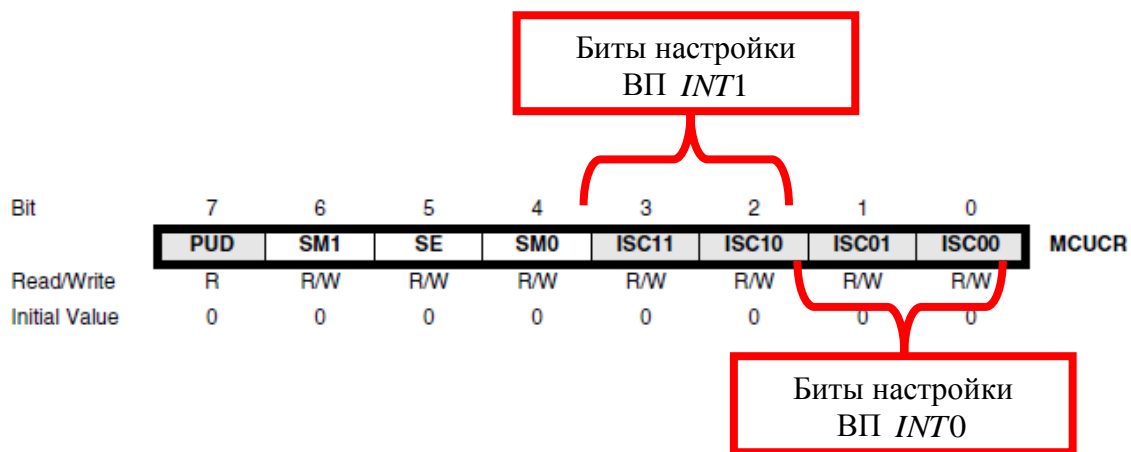
**Рис. 2.** Внешние прерывания микроконтроллера *ATtiny2313*



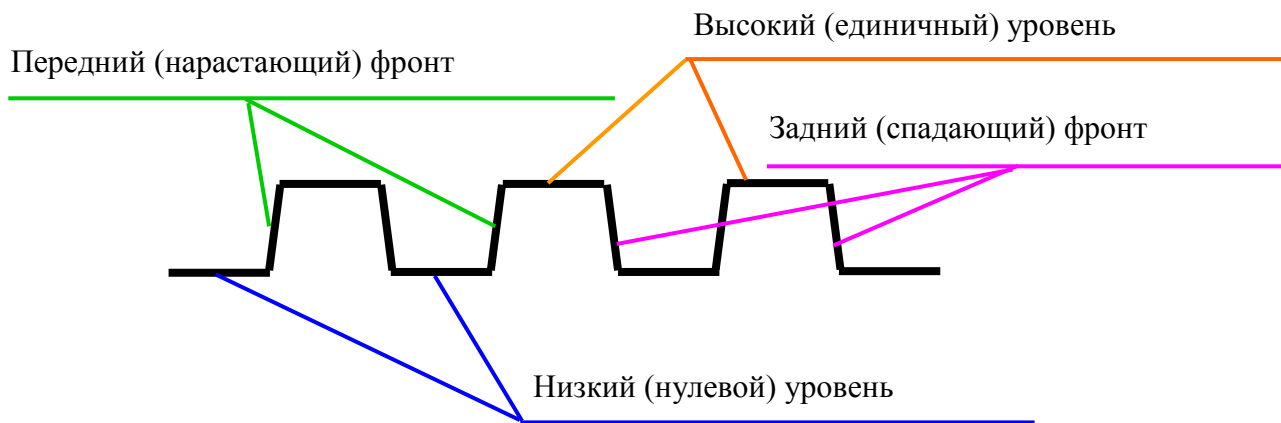
Бит *I* установлен – глобальное разрешение прерываний (язык Си: `sei()`; язык Ассемблера: `sei`)

Бит *I* сброшен – глобальный запрет прерываний (язык Си: `cli()`; язык Ассемблера: `cli`)

**Рис. 3.** Глобальное разрешение и глобальный запрет прерываний в регистре статуса *SREG*



**Рис. 4.** Регистр управления внешними прерываниями микроконтроллера MCUCR



**Рис. 5.** Элементы цифрового сигнала

Таблица 1

*Режимы настройки ВП INT0*

Биты регистра MCUCR		Описание
ISC01	ISC00	
0	0	ВП вызывается по низкому уровню
0	1	ВП вызывается по изменению логического уровня на выводе
1	0	ВП вызывается по заднему (спадающему) фронту
1	1	ВП вызывается по переднему (нарастающему) фронту

Таблица 2

*Режимы настройки ВП INT1*

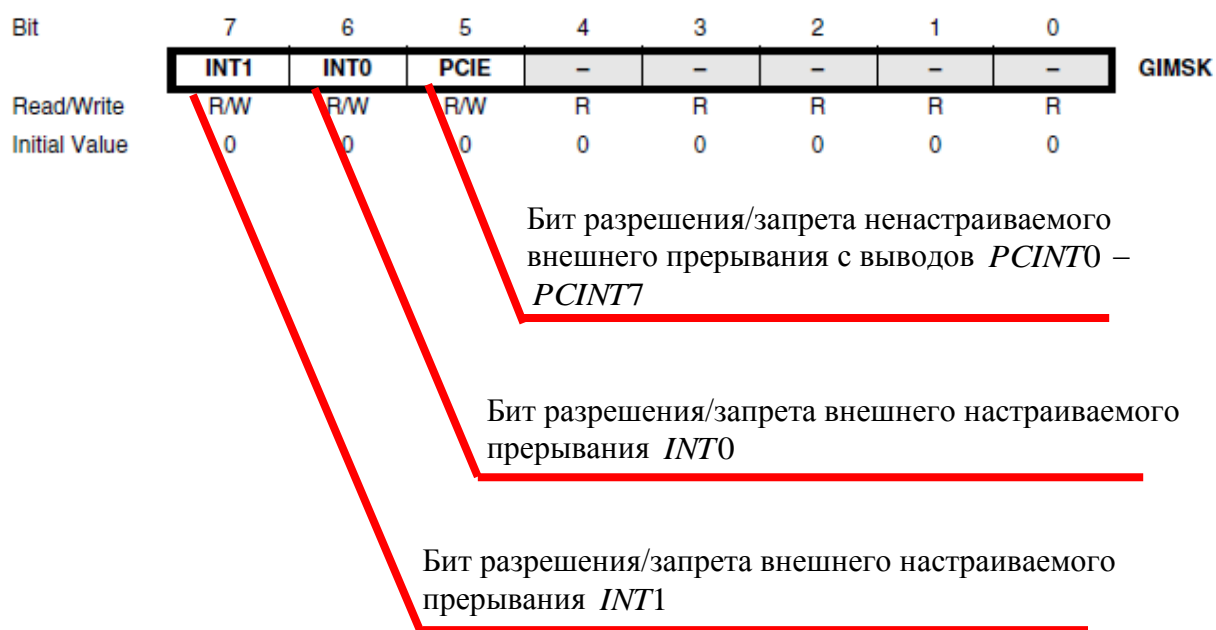
Биты регистра MCUCR		Описание
ISC11	ISC10	
0	0	ВП вызывается по низкому уровню
0	1	ВП вызывается по изменению логического уровня на выводе
1	0	ВП вызывается по заднему (спадающему) фронту
1	1	ВП вызывается по переднему (нарастающему) фронту

Для разрешения ВП используется регистр маски ВП GIMSK (рис. 6):

- Бит 7 (*INT1*) – разрешение/запрет ВП *INT1*. Вызов прерывания на выводе *INT1* разрешен, если бит *INT1* в регистре маски ВП GIMSK и бит *I* в регистре статуса SREG установлены. Биты ISC10 и ISC11 в регистре управления ВП MCUCR определяют, в каком режиме будет считываться сигнал прерывания на выводе *INT1*.

- Бит 6 (*INT0*) – разрешение/запрет настраиваемого ВП *INT0*. Вызов прерывания на выводе *INT0* разрешен, если бит *INT0* в регистре маски внешних прерываний GIMSK и бит *I* в регистре статуса SREG установлены. Биты ISC00 и ISC01 в регистре управления ВП MCUCR определяют, в каком режиме будет считываться сигнал прерывания на выводе *INT1*.

- Бит 5 (*PCIE*) – разрешение/запрет ненастраиваемого ВП на выводах *PCINT0* – *PCINT7*. Ненастраиваемое ВП вызывается по изменению логического уровня на выводах *PCINT0* – *PCINT7*. Вызов прерывания разрешен, если бит *PCIE* в регистре маски внешних прерываний GIMSK и бит *I* в регистре статуса SREG установлены.



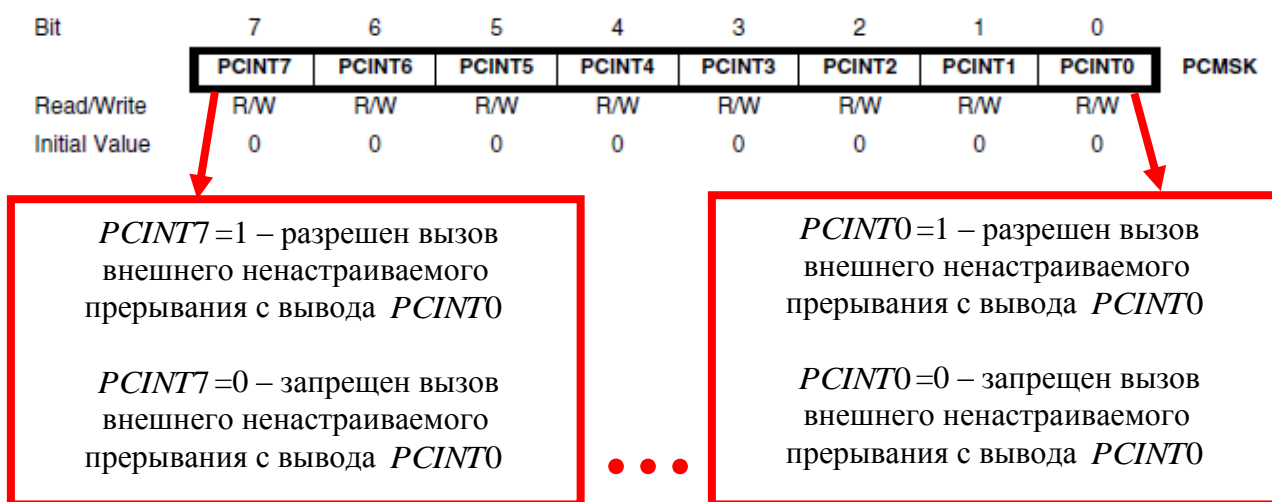
**Рис. 6.** Регистр маски внешних прерываний GIMSK

Какие именно выводы (*PCINT0* – *PCINT7*) будут вызывать ненастраиваемое ВП определяется индивидуально установкой бит в регистре маски ненастраиваемого ВП PCMSK (рис. 7). Каждый бит из *PCINT0* – *PCINT7* отвечает за соответствующий вывод микроконтроллера *ATTiny2313*. Установка какого-либо бита из *PCINT0* – *PCINT7* разрешает соответствующему выводу работать в качестве источника прерывания. Если какой-либо бит из *PCINT0* – *PCINT7* сброшен, то соответствующий вывод не будет работать в качестве источника прерывания.

Пример настройки прерывания *INT0* на языке Си:

```
MCUCR=0b00000011; // для INT0 настроен передний фронт
GIMSK=0b01000000; //разрешения прерывания INT0
```

```
sei(); //глобальное разрешение прерываний по всей программе
```



**Рис. 7.** Регистр маски внешнего ненастраиваемого прерывания *PCINT0*

Биты регистра флагов внешних прерываний EIFR (рис. 8):

- Бит 7 (INTF1) – флаг настраиваемого ВП *INT1*. Когда происходит запрос прерывания на выводе *INT1*, то флаг INTF1 устанавливается. Если бит *INT1* в регистре маски ВП GIMSK и бит *I* в регистре статуса SREG установлены, то при установке флага INTF1 микроконтроллер перейдет к выполнению подпрограммы обработки ВП. Флаг INTF1 очищается аппаратно при запуске процедуры обработки ВП.

- Бит 6 (INTF0) – флаг настраиваемого ВП *INT0*. Когда происходит запрос прерывания на выводе *INT0*, то флаг INTF0 устанавливается. Если бит *INT0* в регистре маски ВП GIMSK и бит *I* в регистре статуса SREG установлены, то при установке флага INTF0 микроконтроллер перейдет к выполнению подпрограммы обработки ВП. Флаг INTF0 очищается аппаратно при запуске процедуры обработки ВП.

- Бит 5 (PCIF) – флаг ненастраиваемого ВП. Изменение логического уровня на одном из выводов *PCINT0* – *PCINT7* генерирует запрос на прерывание, что приводит к установке флага PCIF. Если бит PCIE в регистре маски ВП GIMSK и бит *I* в регистре статуса SREG установлены, то при установке флага PCIF микроконтроллер перейдет к выполнению подпрограммы обработки прерывания. Флаг PCIF очищается аппаратно при запуске процедуры обработки ВП.



**Рис. 8.** Регистр флагов внешних прерываний EIFR

Пример структуры кода на языке Си для обработки настраиваемых ВП:

```
#include <avr/io.h>
#include <avr/interrupt.h> // библиотека прерываний

//подпрограмма обработки прерывания INT0
ISR(INT0_vect)
{
    cli(); //глобальный запрет прерываний

    // код, который должен выполняться
    //при срабатывании прерывания INT0

    sei(); //глобальное разрешение прерываний
}

// подпрограмма обработки прерывания INT1
ISR(INT1_vect)
{
    /*код, который должен выполняться
    при срабатывании прерывания INT1*/
}

int main(void)
{
    //настройка прерываний INT0 и INT1 (передний фронт)
    MCUCR=0b00001111;
    //разрешение прерываний INT0 и INT1
    GIMSK=0b11000000;
    sei(); //глобальное разрешение прерываний

    while(1)
    {
        /*код, который будет выполняться
        в основном цикле*/
    }
}
```

## II ЗАДАНИЕ

Задание 3.4 из лабораторной работы 1-2 выполнить с использованием внешних прерываний.

# Лабораторная работа № 4

## Разработка программы на языке Си, работающей с внешними прерываниями и таймерами

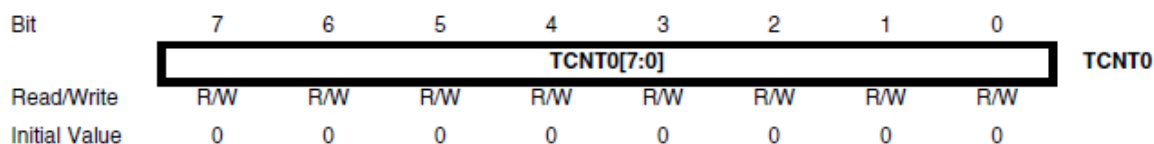
### I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

#### Восьмиразрядный таймер/счетчик МК ATtiny2313

**Таймер/счётчик T0 (T/C T0)** микроконтроллера *ATtiny2313* – это универсальный восьмиразрядный счётный модуль (СМ) с независимыми модулями совпадения (МС) и поддержкой широтно-импульсной модуляции (ШИМ). К основным особенностям T/C T0 можно отнести:

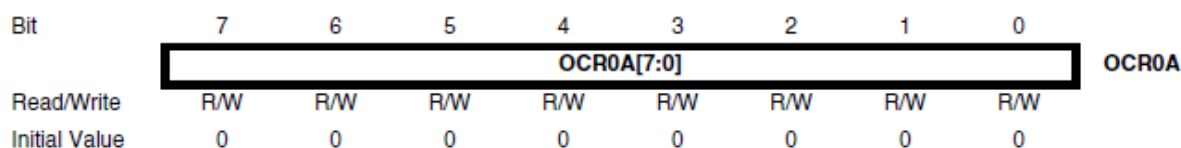
- два независимых МС (МС по каналу *A* и МС по каналу *B*);
- сброс таймера при совпадении (в режимах сброса таймера при совпадении (*CTC*), «Быстрый ШИМ» с предельным значением в регистре сравнения (РС) *OCR0A* и «ШИМ, корректный по фазе» с предельным значением в РС *OCR0A* при совпадении значений счетного регистра (СР) *TCNT0* и РС *OCR0A* происходит сброс значения СР *TCNT0* (*TCNT0* = 0x00));
- программно изменяемый период в ШИМ-режимах (режимы «Быстрый ШИМ» и «ШИМ, корректный по фазе»);
- три независимых источника прерывания (прерывание по совпадению в канале *A*, прерывание по совпадению в канале *B*, прерывание по переполнению).

Основой восьмиразрядного T/C T0 является СМ. Когда T/C T0 запущен, то по каждому импульсу тактового сигнала изменяется значение восьмиразрядного СР *TCNT0* (рис. 1). Установленный режим работы («Normal», сброс при совпадении (*CTC*), «Быстрый ШИМ» или «ШИМ, корректный по фазе») определяет поведение СР *TCNT0* (значения СР *TCNT0* может очищаться, увеличиваться или уменьшаться.) Тактовые импульсы могут быть получены от внешнего или внутреннего источника (определяется битами *CS00* – *CS02* в регистре *B* управления T/C T0 *TCCR0B*).



**Рис. 1.** Счетный регистр T/C T0 – *TCNT0*

Основой модуля совпадения является восьмиразрядный цифровой компаратор, который постоянно сравнивает значение СР *TCNT0* с РС *OCR0A* и РС *OCR0B* (рис. 2, 3). В случае равенства регистров компаратор выдает сигнал, устанавливается соответствующий флаг равенства (для РС *OCR0A* – бит *OCF0A* в регистре флагов *TIFR*; для РС *OCR0B* – бит *OCF0B* в регистре флагов *TIFR*) и могут быть выполнены определенные действия (например, вызов прерывания).



**Рис. 2.** Регистр сравнения T/C T0  
(канал *A*) – *OCR0A*

Bit	7	6	5	4	3	2	1	0	
	OCR0B[7:0]								OCR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Рис. 3.** Регистр сравнения Т/С T0  
(канал B) – OCR0B

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

*Флаг разрешения прерывания  
по совпадению в канале B (OCIE0B).*

Если установлены бит OCIE0B и флаг I регистра  
состояния SREG, то прерывание по совпадению в  
канале B Т/С T0 разрешено

*Флаг разрешения прерывания по переполнению Т/С T0  
(TOIE0).*

Если установлены бит TOIE0 и флаг I регистра  
состояния SREG, то прерывание  
по переполнению Т/С T0 разрешено

*Флаг разрешения прерывания  
по совпадению в канале A (OCIE0A).*

Если установлены бит OCIE0A и флаг I регистра  
состояния SREG, то прерывание по совпадению в канале  
A Т/С T0 разрешено

**Рис. 4.** Регистр маски TIMSK (для Т/С T0)

Статусные флаги, устанавливающиеся при возникновении событий, расположены в регистре флагов Т/С T0 TIFR (рис. 5), являющимся общим регистром для таймеров/счетчиков микроконтроллера ATtiny2313. Для Т/С T0 в регистре TIFR расположены три бита (рис. 5).

Bit	7	6	5	4	3	2	1	0	
	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	TIFR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

*Статусный флаг совпадения в канале В (OCF0B).*

Бит OCF0B устанавливается, когда возникает совпадение содержимого СР TCNT0 и РС OCR0B. Бит OCF0B аппаратно сбрасывается в ноль в тот момент, когда начинается выполнение соответствующей подпрограммы обработки прерывания. Для возникновения прерывания по совпадению в канале В должны быть установлены биты OCIE0B (регистр TIMSK), I (регистр состояния SREG) и флаг OCF0B

*Статусный флаг переполнения Т/С T0 (TOV0).*

Бит TOV0 устанавливается, когда происходит переполнение Т/С T0. Бит TOV0 аппаратно сбрасывается в ноль в тот момент, когда начинается выполнение соответствующей подпрограммы обработки прерывания. Для возникновения прерывания по переполнению должны быть установлены биты TOIE0 (регистр TIMSK), I (регистр состояния SREG) и флаг TOV0

*Статусный флаг совпадения в канале А (OCF0A).*

Бит OCF0A устанавливается, когда возникает совпадение содержимого СР TCNT0 и РС OCR0A. Бит OCF0A аппаратно сбрасывается в ноль в тот момент, когда начинается выполнение соответствующей подпрограммы обработки прерывания. Для возникновения прерывания по совпадению в канале А должны быть установлены биты OCIE0A (регистр TIMSK), I (регистр состояния SREG) и флаг OCF0A

**Рис. 5.** Регистр флагов TIFR (для Т/С T0)

Режимы работы Т/С T0 задаются битами WGM00 - WGM02 в регистрах TCCR0A (нулевой и первый биты) и TCCR0B (третий бит) (рис. 6, 7) в соответствии с табл. 1:

1. Режим «Normal». Направление счета является инкрементирующим. Счетчик достигает максимального значения 0xFF, а затем перезапускается заново со значения 0x00. Флаг переполнения TOV0 устанавливается при достижении счетчиком значения 0xFF. Режим «Normal» задан по умолчанию.

Настройка режима «Normal» на языке Си:

```
TCCR0A = TCCR0A & ~(1<<1) & ~(1<<0);
TCCR0B = TCCR0B & ~(1<<3);
```



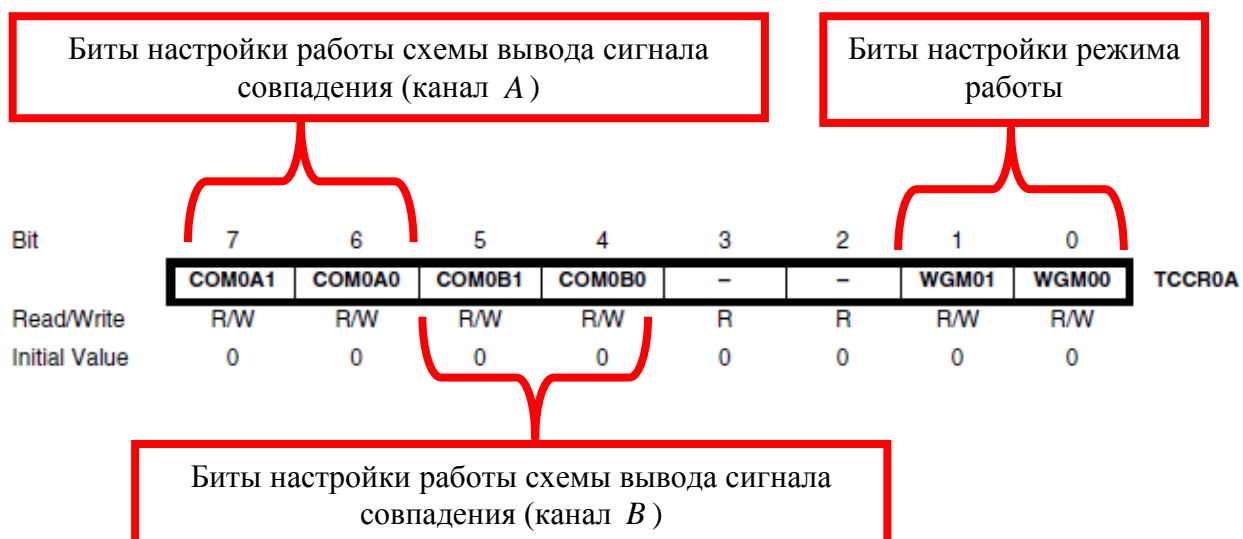


Рис. 6. Регистр A управления T/C T0 – TCCR0A



Рис. 7. Регистр B управления T/C T0 – TCCR0B

Таблица 1

Режимы работы T/C T0

Бит регистра TCCR0B	Биты регистра TCCR0A		Название режима	Верхний предел	Установка флага TOV0
	WGM01	WGM00			
WGM02	WGM01	WGM00			
0	0	0	Normal/	0xFF	0xFF
0	0	1	Phase Correct PWM	0xFF	0x00
0	1	0	CTC	OCR0A	0xFF
0	1	1	Fast PWM	0xFF	0xFF
1	0	1	Phase Correct PWM	OCR0A	0x00
1	1	1	Fast PWM	OCR0A	OCR0A

2. Режим сброса при совпадении (Clear Timer on Compare – *CTC*). Направление счета является инкрементирующим. Значение СР TCNT0 сбрасывается в нуль при достижении значения, записанного в РС OCR0A. Максимальное значение счетчика определяется значением РС OCR0A. Флаг переполнения TOV0 устанавливается при достижении счетчиком значения 0xFF, поэтому для работы с прерыванием по переполнению в РС OCR0A необходимо записать значение 0xFF. Для вызова прерывания по совпадению в канале А необходимо установить бит OCIE0A регистра TIMSK и флаг I регистра состояния SREG. Для вызова прерывания по совпадению в канале В значение РС OCR0B должно быть меньше значения РС OCR0A, а также установлены биты OCIE0A (регистр TIMSK) и I (регистр состояния SREG).

Настройка режима *CTC* на языке Си:

$$TCCR0A = (TCCR0A / (1 << 1)) \& \sim (1 << 0);$$

$$TCCR0B = TCCR0B \& \sim (1 << 3);$$

3. Режим «Быстрый ШИМ» (FastPWM). Направление счета является инкрементирующим. Возможны следующие настройки:

1) максимальное значения СР TCNT0 = 0xFF (настройка на языке Си):

$$TCCR0A = TCCR0A / (1 << 0) / (1 << 1);$$

$$TCCR0B = TCCR0B \& \sim (1 << 3);$$

Флаг переполнения TOV0 устанавливается при достижении СР TCNT0 значения 0xFF;

2) максимальное значения СР TCNT0 определяется значением РС OCR0A (настройка на языке Си):

$$TCCR0A = TCCR0A / (1 << 0) / (1 << 1);$$

$$TCCR0B = TCCR0B / (1 << 3);$$

Флаг переполнения TOV0 устанавливается при достижении СР TCNT0 значения, записанного в РС OCR0A.

В режиме «Быстрый ШИМ» для вызова прерывания по совпадению в канале А необходимо записать значение в РС OCR0A, установить бит OCIE0A регистра TIMSK и флаг I регистра состояния SREG. Для вызова прерывания по совпадению в канале В необходимо записать значение в РС OCR0B, установить биты OCIE0B (регистр TIMSK) и I (регистр состояния SREG). Если прерывание по совпадению в канале В вызывается в режиме «Быстрый ШИМ» с максимальным значением счетчика, определяемым значением РС OCR0A, то значение РС OCR0B должно быть меньше значения РС OCR0A.

4. Режим «ШИМ, корректный по фазе» (PhaseCorrectPWM). Счетчик периодически изменяет направление своего счета (от минимального до максимального значения, затем от максимального до минимального, далее направление счета снова меняется и все повторяется сначала). Возможны следующие настройки:

1) максимальное значения СР TCNT0 = 0xFF (настройка на языке Си):

$$TCCR0A = (TCCR0A \& \sim (1 << 1)) / (1 << 0);$$

$$TCCR0B = TCCR0B \& \sim (1 << 3);$$

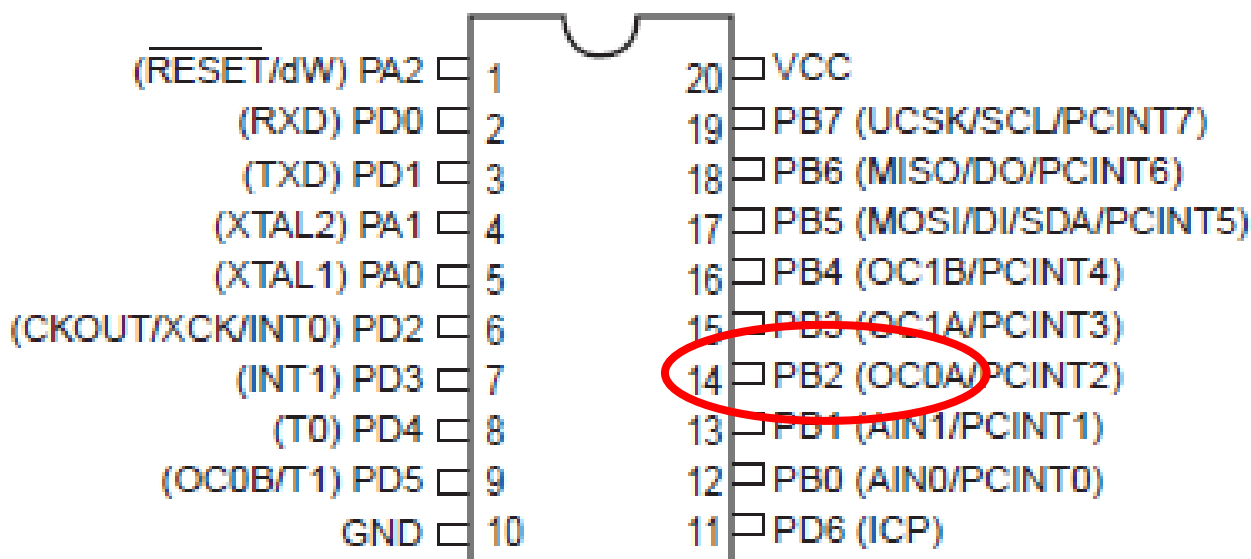
2) максимальное значения СР TCNT0 определяется значением РС OCR0A (настройка на языке Си):

$$TCCR0A = (TCCR0A \& \sim (1 < 1)) / (1 < 0);$$

$$TCCR0B = TCCR0B / (1 < 3);$$

В режиме «ШИМ, корректный по фазе» флаг переполнения TOV0 устанавливается при достижении счетчиком значения 0x00. Для вызова прерывания по совпадению в канале A необходимо записать значение в PC OCR0A, установить бит OCIE0A регистра TIMSK и флаг I регистра состояния SREG. Для вызова прерывания по совпадению в канале B необходимо записать значение в PC OCR0B, установить биты OCIE0B (регистр TIMSK) и I (регистр состояния SREG). Если прерывание по совпадению в канале B вызывается в режиме «ШИМ, корректный по фазе» с максимальным значением счетчика, определяемым значением PC OCR0A, то значение PC OCR0B должно быть меньше значения PC OCR0A.

Для настройки работы схемы вывода сигнала совпадения по каналу A используются биты COM0A1 и COM0A0 регистра TCCR0A (см. рис. 6). Генерация сигнала осуществляется с вывода OC0A (PB2) (рис. 8). Для режимов «Normal» и CTC настройка схемы вывода сигнала совпадения представлена в табл. 2, для режима «Быстрый ШИМ» – в табл. 3, а для режима «ШИМ, корректный по фазе» – в табл. 4.



**Рис. 8.** Схема микроконтроллера ATtiny2313 (вывод OC0A для вывода сигнала совпадения по каналу A)

Таблица 2

*Настройки работы схемы вывода сигнала совпадения по каналу A для режимов «Normal» и CTC*

Биты регистра TCCR0A		Описание
COM0A1	COM0A0	
0	0	Стандартный режим порта. Выход OC0A не подключен
0	1	Переключение OC0A на противоположное в момент совпадения
1	0	Сброс OC0A в момент совпадения
1	1	Установка OC0A в момент совпадения

Таблица 3

*Настройки работы схемы вывода сигнала совпадения  
по каналу А для режима «Быстрый ШИМ»*

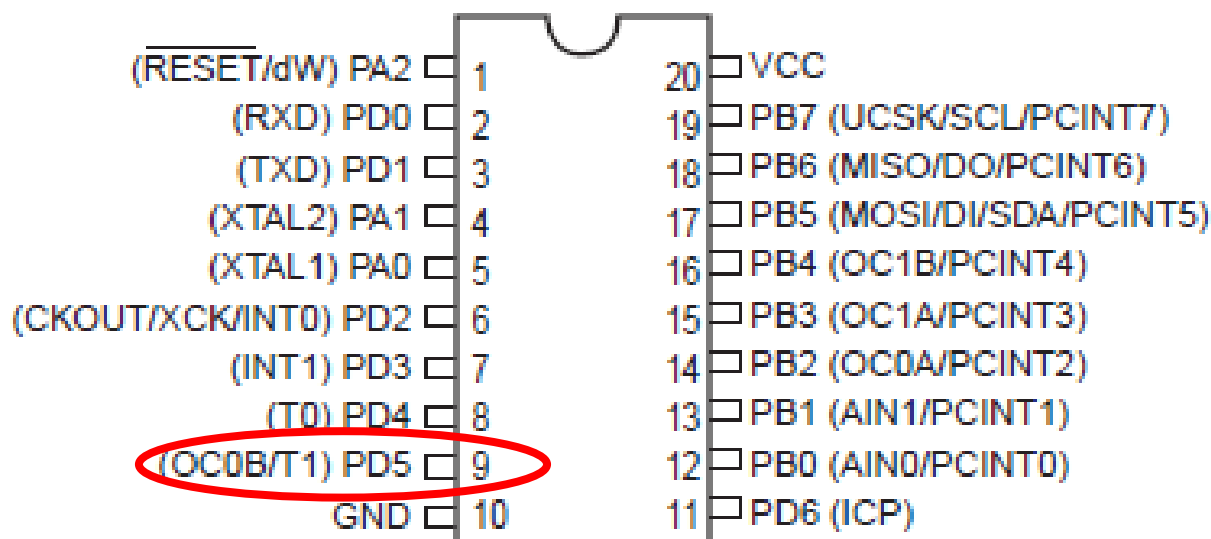
Биты регистра TCCR0A		Описание
COM0A1	COM0A0	
0	0	Стандартный режим порта. Выход OC0A не подключен
0	1	WGM02=0 : Стандартный режим порта. Выход OC0A не подключен. WGM02=1 : Переключение OC0A в момент совпадения
1	0	Сброс OC0A в момент совпадения, установка OC0A при достижении счетчиком максимального значения
1	1	Установка OC0A в момент совпадения, сброс при достижении счетчиком максимального значения

Таблица 4

*Настройки работы схемы вывода сигнала совпадения  
по каналу А для режима «ШИМ, корректный по фазе»*

Биты регистра TCCR0A		Описание
COM0A1	COM0A1	
0	0	Стандартный режим порта. Выход OC0A не подключен
0	1	WGM02=0 : Стандартный режим порта. Выход OC0A не подключен. WGM02=1 : Переключение OC0A в момент совпадения
1	0	Сброс OC0A в момент совпадения при прямом счете. Установка OC0A в момент совпадения при обратном счете
1	1	Установка OC0A в момент совпадения при прямом счете. Сброс OC0A в момент совпадения при обратном счете

Для настройки работы схемы вывода сигнала совпадения по каналу В используются биты COM0B1 и COM0B0 регистра TCCR0A (см. рис. 6). Генерация сигнала осуществляется с вывода OC0B (PD5) (рис. 9) Для режимов «Normal» и CТС настройка схемы вывода сигнала совпадения представлена в табл. 5, для режима «Быстрый ШИМ» – в табл. 6, а для режима «ШИМ, корректный по фазе» – в табл. 7. Для режимов CТС, «Быстрый ШИМ» с предельным значением, определяемым PC OCR0A, и «ШИМ, корректный по фазе» с предельным значением, определяемым в PC OCR0A, значение PC OCR0B должно быть меньше значения PC OCR0A.



**Рис. 9.** Схема микроконтроллера *ATtiny2313* (вывод *OC0B* для вывода сигнала совпадения по каналу *B*)

Таблица 5

*Настройки работы схемы вывода сигнала совпадения по каналу B для режимов «Normal» и CTC*

Биты регистра TCCR0A		Описание
COM0B1	COM0B0	
0	0	Стандартный режим порта. Выход <i>OC0B</i> не подключен
0	1	Переключение <i>OC0B</i> в момент совпадения
1	0	Сброс <i>OC0B</i> в момент совпадения
1	1	Установка <i>OC0B</i> в момент совпадения

Таблица 6

*Настройки работы схемы вывода сигнала совпадения по каналу B для режима «Быстрый ШИМ»*

Биты регистра TCCR0A		Описание
COM0B1	COM0B0	
0	0	Стандартный режим порта. Выход <i>OC0B</i> не подключен
0	1	Зарезервировано
1	0	Сброс <i>OC0B</i> в момент совпадения, установка <i>OC0B</i> при достижении счетчиком максимального значения
1	1	Установка <i>OC0B</i> в момент совпадения, сброс при достижении счетчиком максимального значения

Таблица 7

*Настройки работы схемы вывода сигнала совпадения  
по каналу В для режима «ШИМ, корректный по фазе»*

Биты регистра TCCR0A		Описание
COM0B1	COM0B0	
0	0	Стандартный режим порта. Выход OC0B не подключен
0	1	Зарезервировано
1	0	Сброс OC0B в момент совпадения при прямом счете, установка OC0B в момент совпадения при обратном счете
1	1	Установка OC0B в момент совпадения при прямом счете, сброс OC0B в момент совпадения при обратном счете

Для настройки режима работы тактового генератора используются биты CS00 - CS02 регистра В управления T/C T0 TCCR0B (см. рис. 7). Возможные значения бит CS00 - CS02 представлены в табл. 8.

Настройка T/C T0 на предварительный делитель 1024 на языке Си:

$$TCCR0B = (TCCR0B \& \sim (1 < 1)) / (1 < 0) / (1 < 2);$$

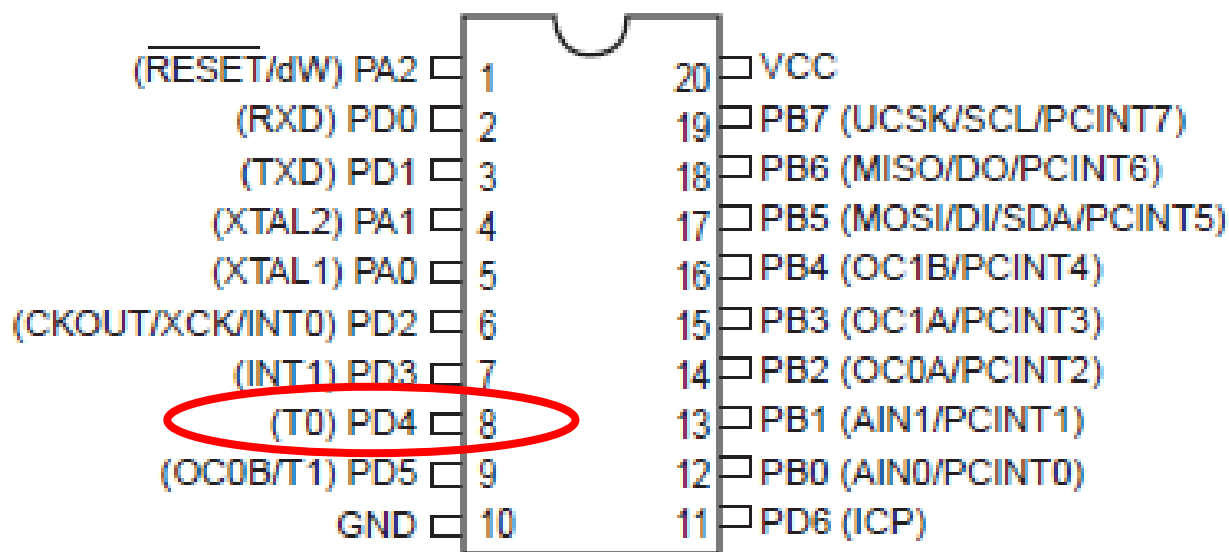
Настройки T/C T0 на внешний источник сигнала с вывода T0 (рис. 10) с синхронизацией по переднему фронту на языке Си:

$$TCCR0B = TCCR0B / (1 < 0) / (1 < 1) / (1 < 2);$$

Таблица 8

*Выбор режима работы тактового генератора T/C T0*

Биты регистра TCCR0B			Описание
CS02	CS01	CS00	
0	0	0	Нет источника сигнала (T/C T0 остановлен)
0	0	1	Тактовая частота МК / 1 (нет предварительного делителя, T/C T0 работает)
0	1	0	Тактовая частота МК / 8 (T/C T0 запущен)
0	1	1	Тактовая частота МК / 64 (T/C T0 запущен)
1	0	0	Тактовая частота МК / 256 (T/C T0 запущен)
1	0	1	Тактовая частота МК / 1024 (T/C T0 запущен)
1	1	0	Внешний источник сигнала, синхронизация по заднему фронту (вход T0, рис. 10)
1	1	1	Внешний источник сигнала, синхронизация по переднему фронту (вход T0, рис. 10)



**Рис. 10.** Схема микроконтроллера *ATTiny2313*  
(вывод T0 для получения тактовых импульсов для Т/С T0  
от внешнего источника)

## II ЗАДАНИЕ

Задание 3.4 из лабораторной работы 1-2 выполнить с использованием внешних прерываний и таймеров.

При выполнении лабораторной работы режимы работы гирлянды реализовать различными режимами работы таймера/счетчика.

Один из режимов работы гирлянды реализовать с использованием схемы вывода сигнала совпадения.

# Лабораторная работа № 5

## Разработка программы на языке Си, работающей с приемопередатчиком (USART)

### I ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

#### Универсальный синхронно-асинхронный последовательный приемопередатчик (USART) МК ATTiny2313

Универсальный синхронно-асинхронный последовательный приемопередатчик (USART) – устройство последовательной передачи информации.

Особенности:

- полно-дуплексная организация (независимые регистраторы последовательного приема и передачи);
- синхронный и асинхронный режимы работы;
- выбор скорости передачи информации;
- поддержка кадров длиной 5-9 бит и 1-2 стоп-бита;
- аппаратная поддержка генерации и проверки сигнала четности;
- обнаружение переполнения данных;
- обнаружение ошибок кадрирования;
- режим межпроцессорных связей;
- двухскоростной режим асинхронной передачи.

Единицей передачи данных является кадр.

Кадр – одно слово данных и сопутствующие ему биты синхронизации (стартовый бит, стоповые биты). Сюда же может быть добавлен бит четности, который применяется для проверки правильности передачи информации.

Поддерживается 30 разных вариантов формата кадров. Любой допустимый формат имеет следующие элементы (рисунок 1):

- один стартовый бит (всегда «0»);
- 5, 6, 7, 8 или 9 бит данных;
- бит четности (если включен контроль четности);
- один или два стоп-бита (всегда «1»).

Самый старший разряд передается последним.

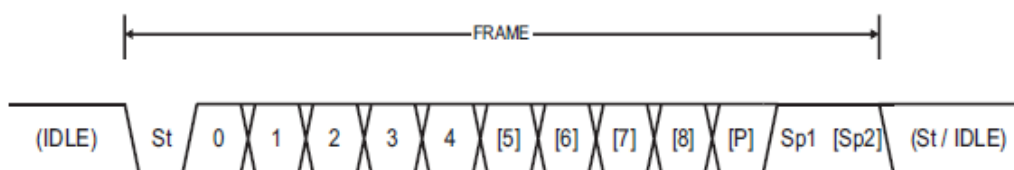


Рис. 1. Формат кадра

#### Инициализация USART:

1. Установка скорости передачи информации.
2. Установка формата кадра.
3. Включение передатчика или приемника в зависимости от выполняемой операции.

Флаг глобального разрешения прерываний должен быть сброшен до окончания процесса инициализации.



### Регистр ввода-вывода данных - UDR

Регистр буфера передаваемых данных и регистр буфера принимаемых данных совместно используют один и тот же адрес в пространстве ввода-вывода, воспринимаемый как единый регистр данных UDR (рисунок 2).

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Рис. 2.** Регистр ввода-вывода данных - UDR

При записи данных в регистр UDR данные попадают в буфер передачи (TXB). При чтении из регистра UDR возвращается содержимое буфера приема (RXB).

При 5-, 6- и 7-битовых режимах работы старшие неиспользуемые разряды передатчика будут игнорироваться, а приемником будут установлены в ноль.

Запись в буфер передачи может производиться только когда флаг UDRE регистра UCSRA установлен. Если данные загружены в буфер передачи и передача разрешена, то передатчик загружает данные в передающий сдвиговый регистр, если он пуст. Далее данные начинают побитово передаваться на выход TXD.

### Регистр статуса и управления «А» - UCSRA

Регистр статуса и управления «А» (UCSRA) представлен на рисунке 3.

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

**Рис. 3.** Регистр статуса и управления «А» (UCSRA)

Описание бит регистра статуса и управления «А» (UCSRA):

*Бит 7 – RXC : Флаг завершения приема.*

Флаг устанавливается, если в буфере приемника есть непрочитанные данные. Флаг очищается, когда буфер приемника пуст. Флаг может быть использован для вызова прерывания по событию «Прием завершен».

*Бит 6 – TXC : Флаг завершения передачи.*

Флаг устанавливается, если очередной кадр в сдвиговом регистре передатчика был полностью передан, а в буфере передатчика (UDR) нет никаких новых данных, предназначенных для передачи. Флаг очищается, когда начинается выполнение соответствующей процедуры обработки прерывания. Флаг может быть использован для вызова прерывания по событию «Передача завершена».

*Бит 5 – UDRE : Флаг «Регистр данных пуст».*

Флаг указывает на готовность буфера передачи (UDR) принимать новые данные. Если флаг установлен в «1», то буфер пуст и готов к записи новых данных. Флаг может быть использован для вызова прерывания по событию «Регистр данных пуст».

*Бит 4 – FE : Флаг ошибки кадрирования.*

Бит устанавливается, если очередная принятая посылка в буфере имеет ошибку кадрирования, то есть если первый стоповый бит посылки в буфере приема оказался

нулевым. Флаг равен нулю, если стоповый бит в принятом кадре равен единице. При перезаписи регистра UCSRA бит FE рекомендуется устанавливать в «0».

*Бит 3 – DOR : Флаг переполнения.*

Бит устанавливается, если обнаружено переполнение данных. Переполнение данных происходит в том случае, когда буфер приема полон, в приемном сдвиговом регистре находится ещё одна посылка и обнаружен новый стартовый бит. Флаг сохраняет свое значение до тех пор, пока приемный буфер (UDR) не будет прочитан. При перезаписи регистра UCSRA бит DOR рекомендуется устанавливать в «0».

*Бит 2 – UPE : Флаг ошибки контроля четности.*

Флаг устанавливается, если очередное слово данных, находящееся в приемном буфере, имеет ошибку четности и проверка четности в момент приема этого слова была разрешена (UPM1 = 1). Флаг действителен до тех пор, пока не прочитан буфер приема. При перезаписи регистра UCSRA бит UPE рекомендуется устанавливать в «0».

*Бит 1 – U2X : Удвоение скорости обмена.*

Бит используется только в асинхронном режиме работы. При работе в синхронном режиме данный бит рекомендуется устанавливать в «0». При установке бита уменьшается коэффициент деления делителя в формирователе скорости обмена, что приводит к удвоению скорости передачи/приема информации.

*Бит 0 – MPCM : Режим мультипроцессорного обмена.*

Бит включает режим мультипроцессорного обмена. При установке бита, все входящие кадры, полученные приемником и не являющиеся адресом, будут игнорироваться. Установка бита не влияет на работу передатчика.

## Регистр статуса и управления «В» - UCSRB

Регистр статуса и управления «В» (UCSRB) представлен на рисунке 4.

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Рис. 4.** Регистр статуса и управления «В» (UCSRB)

Описание бит регистра статуса и управления «В» (UCSRB):

*Бит 7 – RXCIE : Разрешение прерывания по завершению приема.*

При установке бита разрешается генерация прерывания при установке флага RXC. Прерывание будет сгенерировано, если установлены бит RXCIE, бит I регистра SREG и бит RXC регистра UCSRA.

*Бит 6 – TXCIE : Разрешение прерывания по завершению передачи.*

При установке бита разрешается генерация прерывания при установке флага TXC. Прерывание будет сгенерировано, если установлены бит TXCIE, бит I регистра SREG и бит TXC регистра UCSRA.

*Бит 5 – UDRIE : Разрешение прерывания по событию «Регистр данных пуст».*

При установке бита разрешается генерация прерывания при установке флага UDRE. Прерывание будет сгенерировано, если установлены бит UDRIE, бит I регистра SREG и бит UDRE регистра UCSRA.

*Бит 4 – RXEN : Разрешение приема.*

При установке бита разрешается работа приемника.

*Бит 3 – TXEN : Разрешение передачи.*

При установке бита разрешается работа передатчика.

*Бит 2 – UCSZ2 : Формат посылок.*

*Бит 1 – RXB8 : Восьмой разряд приемного буфера.*

Предназначен для хранения девятого информационного разряда принимаемого слова данных (кадр в 9 разрядов). Бит должен быть прочитан до того, как будет прочитан буфер UDR.

*Бит 0 – TXB8 : Восьмой разряд буфера передачи.*

Девятый информационный разряд слова данных, предназначенный для передачи кадра в 9 бит. Разряд должен быть записан перед тем, как младшие 8 разрядов будут записаны в регистр UDR.

### Регистр статуса и управления «С» - UCSRC

Регистр статуса и управления «С» (UCSRC) представлен на рисунке 5.

Bit	7	6	5	4	3	2	1	0	
	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

**Рис. 5.** Регистр статуса и управления «С» (UCSRC)

Описание бит регистра статуса и управления «С» (UCSRC):

*Бит 6 – UMSEL : Выбор режима работы.*

«0» - асинхронный режим; «1» - синхронный режим.

*Биты 5:4 – UPM1:0 : Выбор режима контроля четности.*

Если контроль четности включен, передатчик автоматически генерирует и посылает биты контроля четности в каждом кадре переданных данных. Приемник генерирует значение четности в каждом кадре для входных данных и сравнивает полученное значение со значением флага UPM0. Если обнаружено несоответствие, то устанавливается флаг UPE регистра UCSRA. Значения бит UPM1:0 для включения проверки на четность или нечетность представлены в таблице 1.

*Бит 3 – USBS : Выбор количества стоповых бит.*

Разряд позволяет выбирать количество стоповых бит, которые будут вставлены передатчиком в конец каждой посылки:

- «0» - один бит;

- «1» - два бита.

*Биты 2:1 – UCSZ1:0 : Формат посылки.*

Биты UCSZ2:0 (UCSZ2 из регистра UCSRB) определяют количество информационных разрядов в кадре (таблица 2).

*Бит 0 – UCPOL : Полярность тактового сигнала.*

Бит используется только в синхронном режиме. При записи в регистр нового значения бит рекомендуется устанавливать в «0». Бит устанавливает связь между фронтами тактового сигнала (ХСК) и моментами передачи/приема очередного бита информации (таблица 3).

**Таблица 1 - Выбор режима контроля четности**

UPM1	UPM0	Режим контроля четности
0	0	Отключено
0	1	Зарезервировано
1	0	Включено, проверка на четность
1	1	Включено, проверка на нечетность

Таблица 2 – Выбор формата посылки

UCSZ2	UCSZ1	UCSZ0	Размер посылки
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	1	1	9 бит

Таблица 3 – Выбор полярности тактового сигнала

UCPOL	Момент передачи очередного бита данных (на выход TXD)	Момент приема очередного бита данных (на вход RXD)
0	Передний фронт сигнала ХСК	Задний фронт сигнала ХСК
1	Задний фронт сигнала ХСК	Передний фронт сигнала ХСК

### Регистр скорости приема/передачи информации – UBRR

Регистр скорости обмена информации UBRR (UBRRL и UBRRH) представлен на рисунке 6. Настройка скорости приема/передачи информации представлена в таблице 4.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Рис. 6. Регистр скорости приема/передачи информации – UBRR

Таблица 4 – Выбор скорости приема/передачи информации при тактовой частоте 1 МГц

Baud Rate (bps)	$f_{osc} = 1.0000 \text{ MHz}$			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%
4800	12	0.2%	25	0.2%
9600	6	-7.0%	12	0.2%
14.4k	3	8.5%	8	-3.5%
19.2k	2	8.5%	6	-7.0%
28.8k	1	8.5%	3	8.5%
38.4k	1	-18.6%	2	8.5%

## Примеры программ

1. Программа, демонстрирующая прием данных:

```
#include <avr/io.h>

int main(void)
{
    DDRB = 0xFF;
    PORTB = 0xFF;

    UCSRB = 0b00010000;
    UBRR = 6;

    while(1)
    {
        while(!(UCSRA & (1 << RXC)));
        PORTB = UDR;
    }
}
```

2. Программа, демонстрирующая передачу данных:

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

int main(void)
{
    UCSRB=0b00001000;
    UBRR=0x06;

    while(1)
    {
        while(!(UCSRA & (1<<UDRE)));
        UDR = 3;
        _delay_ms(200);
    }
}
```

3. Программа, демонстрирующая работу прерывания по завершению приема данных:

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR(USART_RX_vect)
{
    PORTB=UDR;
}
```

```

int main(void)
{
    UCSRB=0b10010000;
    UBRR=0x06;

    DDRB  = 0xFF;
    PORTB = 0xFF;

    sei();

    while(1)
    {
    }
}

```

4. Программа, демонстрирующая работу прерывания по событию «Регистр данных пуст» (передача символа и отключение прерывания):

```

#include <avr/io.h>
#include <avr/interrupt.h>

ISR(USART_UDRE_vect)
{
    UDR = 'A';
    UCSRB &= ~(1<<5);
}

int main(void)
{
    UCSRB = 0b00101000;
    UBRRH = 0x06;

    sei();

    while(1)
    {
    }
}

```

## II ЗАДАНИЕ

3.1. Разработать программу для микроконтроллера на языке Си для управления гирляндой (из лабораторной работы 4) с терминала.

3.2. Разработать терминал, позволяющий управлять гирляндой.