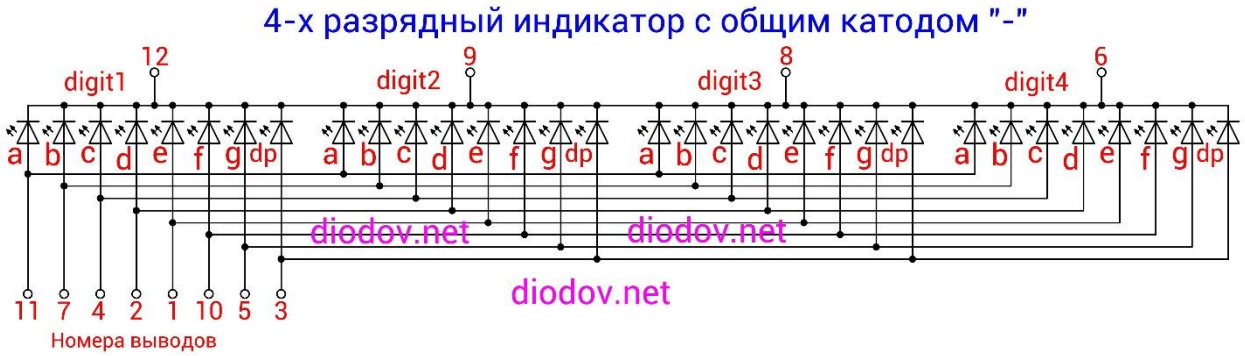
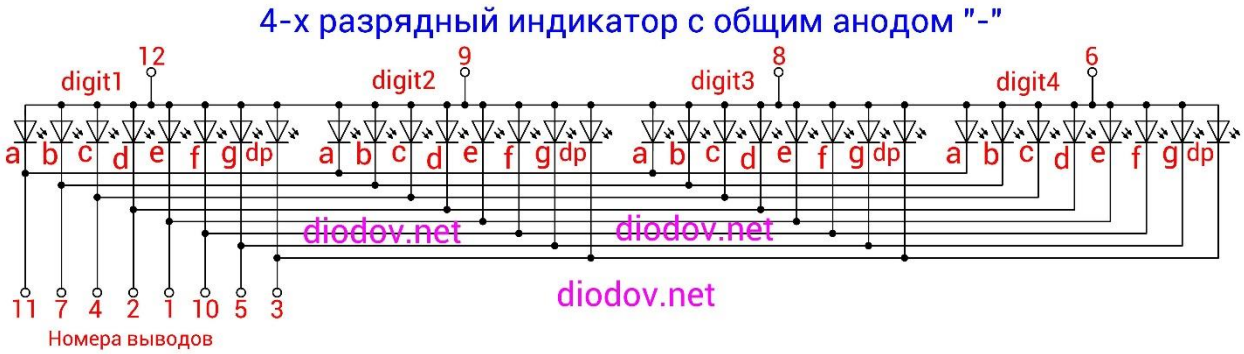
Динамическая индикация на микроконтроллерах.

Динамическая индикация широко применяется для отображения различной информации, например, температуры, напряжения, времени или просто количества срабатывания каких-либо устройств или датчиков. Динамическая индикация на базе семисегментных индикаторов отлично согласуется в совместной работе с микроконтроллерами. Однако в литературе по программированию микроконтроллеров AVR данный вопрос рассмотрен очень поверхностно и далеко не в каждой книге, посвященной соответствующей тематике. Поэтому более подробно рассмотрим, как подключить семисегментный индикатор с динамической индикацией к микроконтроллеру, в данном случае – к ATmega8, но аналогия сохраняется для МК AVR любой серии.

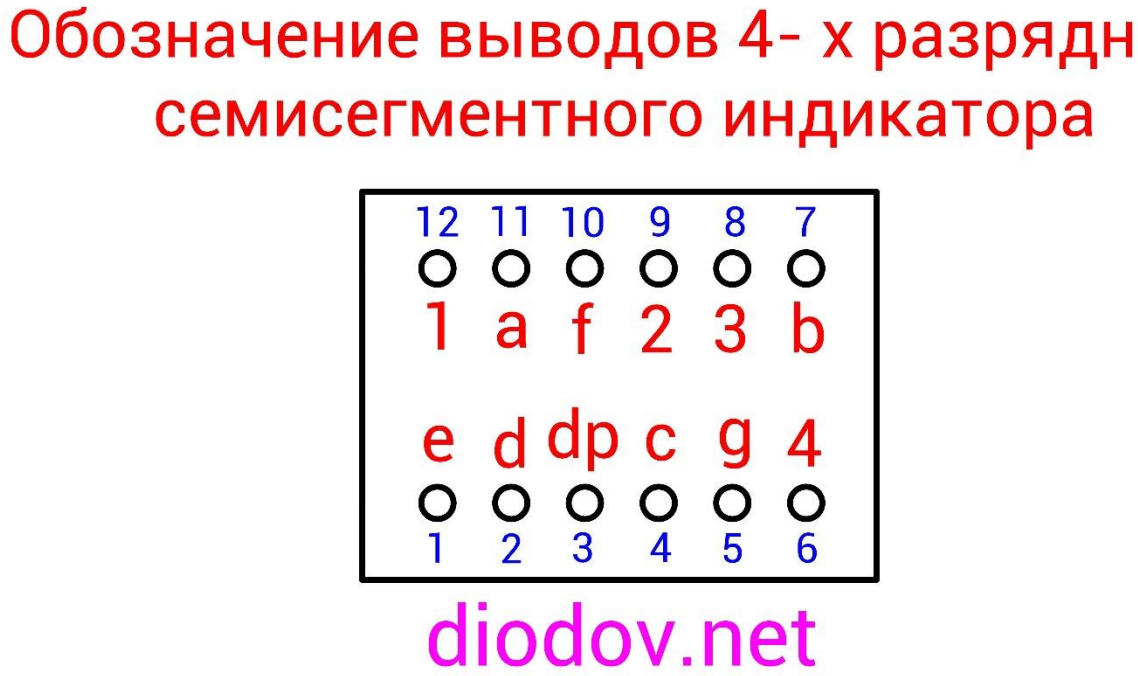
По количеству разрядов (цифр) динамические семисегментные индикаторы бывают одноразрядные, двухразрядные, трехразрядные, четырехразрядные и очень редко – шестиразрядные. Основное внимание мы уделим четырехразрядным семисегментным индикаторам, как наиболее применяемому типу динамической индикации. Изготовляются они с общим анодом и общим катодом. Схемы соединения светодиодов отдельных сегментов представлены на рисунках.



Как видно из рисунков, каждый разряд, называемый digit, имеет свой отдельный общий в пределах разряда вывод. Поскольку рассматривается 4-х разрядная динамическая индикация, то таких выводы четыре – digit1, digit2, digit3, digit4.

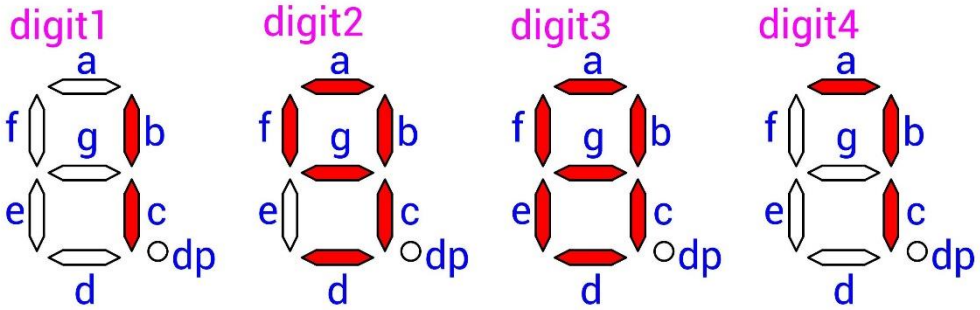


Распиновка выводов 4-х разрядного семисегментного индикатора приведена на рисунке ниже. В данном случае показан вид сверху, то есть индикатор не нужно переворачивать вверх ногами.

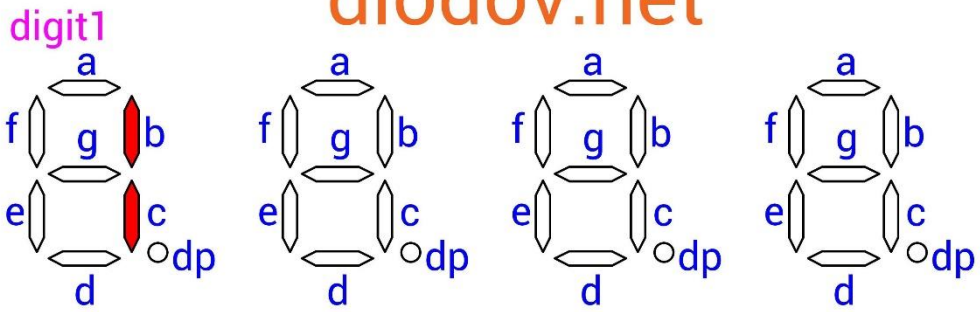


Как работает динамическая индикация

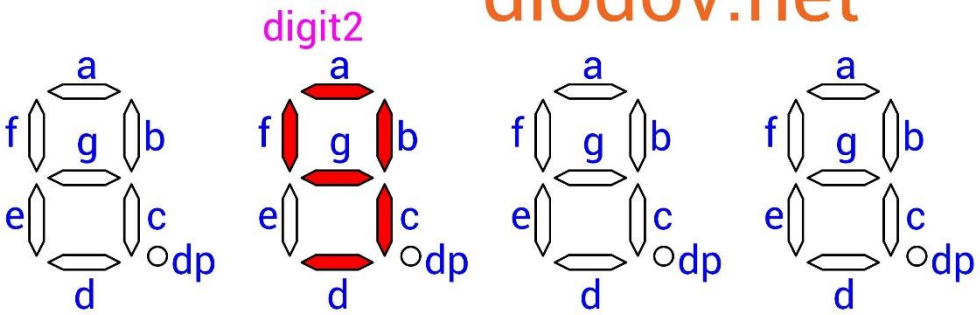
Теперь рассмотрим, как работает динамическая индикация с общим катодом. Например, нам необходимо отобразить число 1987. Для этого следует в первый момент времени подать высокий потенциал на аноды сегментов, образующих единицу – b и c, а на общий катод первого разряда подать низкий потенциал. Общие катоды оставшихся трех разрядов – digit2, digit3 и digit4 остаются не подключенными.



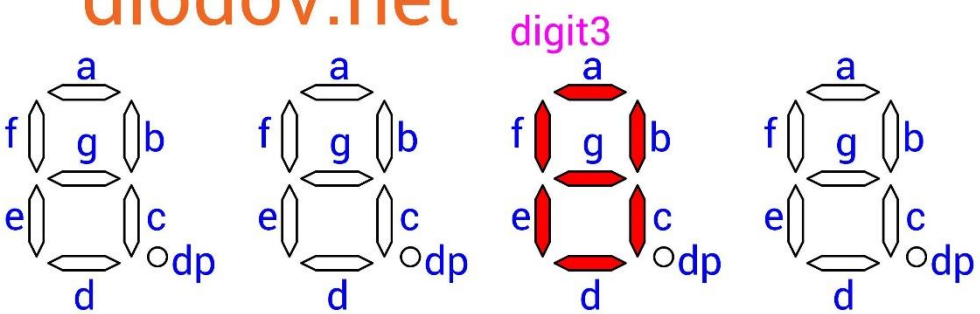
1



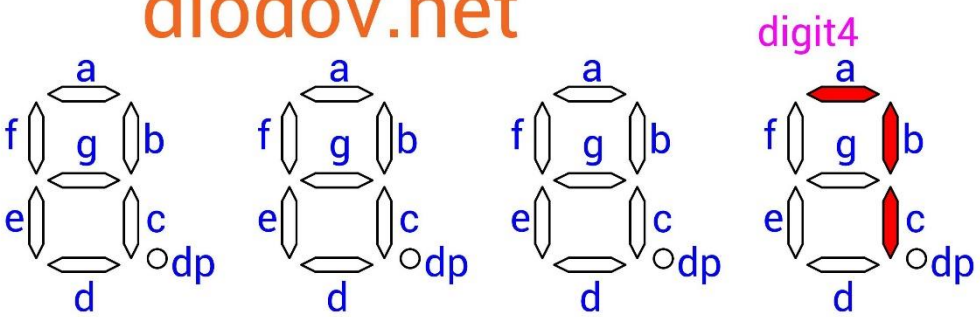
2



3



4



Во второй момент времени получают питания сегменты, образующие цифру 9, общий катод второго разряда подключается к минусу, а digit1 теряет питание; digit2, digit3, как и раннее – остаются не подключенными.

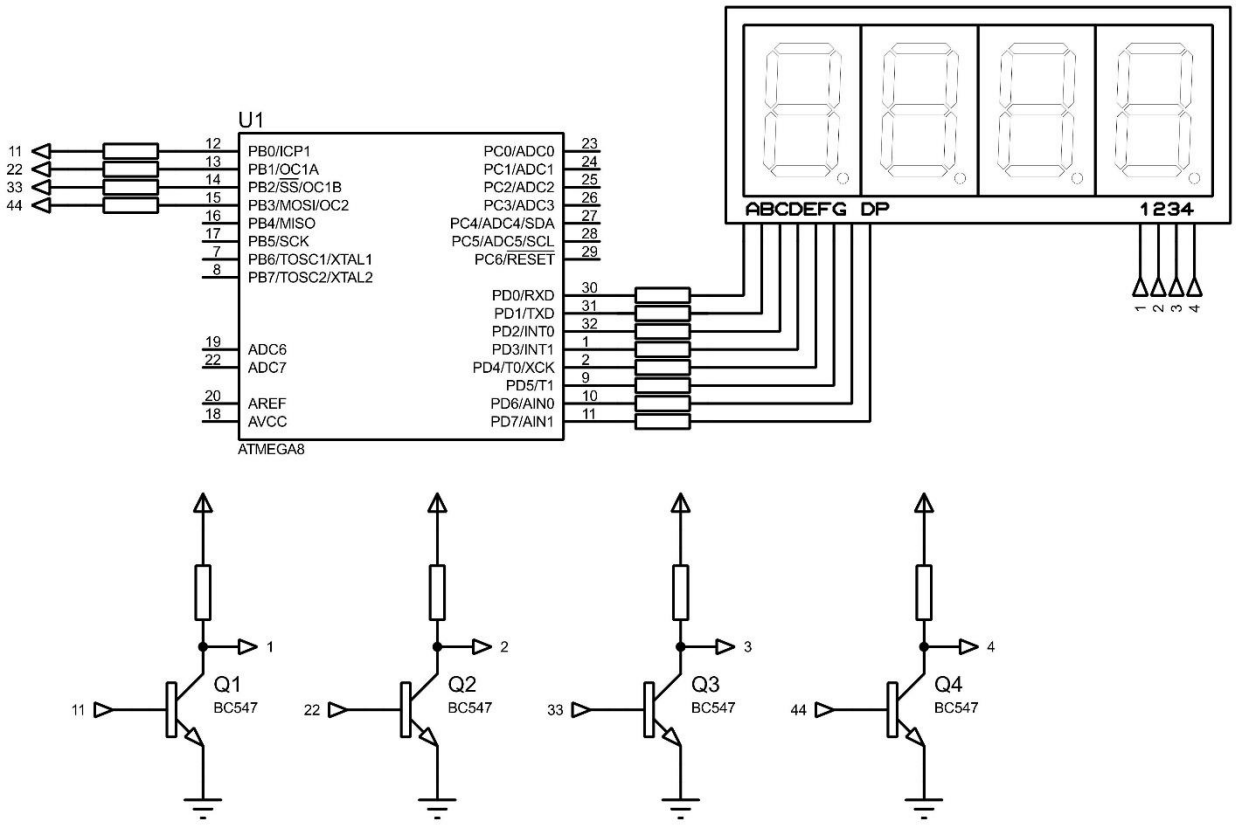
В третий момент времени засвечивается цифра 8 на третьем индикаторе, а остальные индикаторы гаснут.

В четвертый момент времени получает питание последний индикатор и отображается цифра 7.

Далее все повторяется снова. При частоте переключений из разряда на разряда более 25 Гц за счет световой инерции светодиодов наши глаза не успевают заметить, как происходят переключения, поэтому визуально мы воспринимаем целостное свечение одновременно все разрядов.

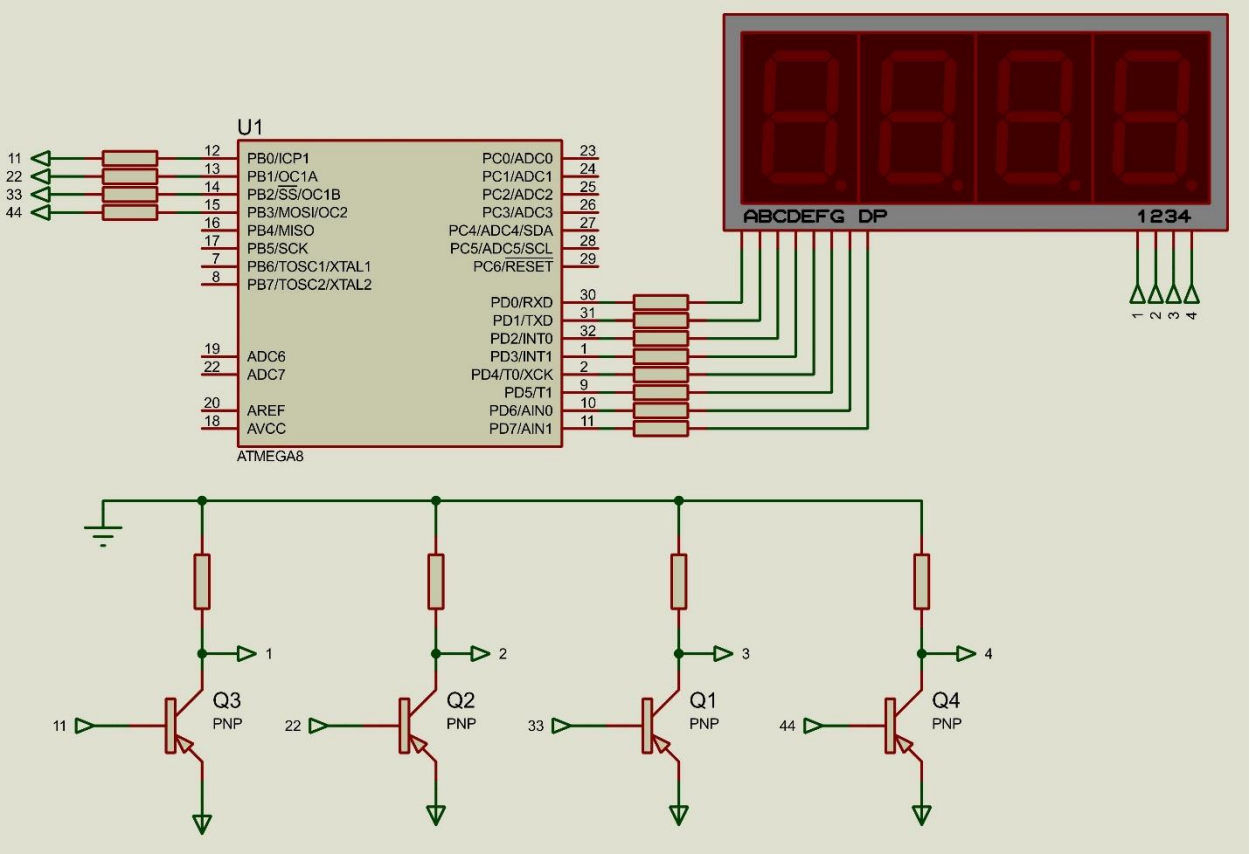
Схема подключения динамической индикации к микроконтроллеру ATmega8

Сегменты динамической индикации будем подключать через токоограничивающие резисторы номиналом 330 Ом к выводам порта D микроконтроллера ATmega8. Выводы, отвечающие digit1, digit2, digit3, digit4 подсоединим через транзисторы n-p-n тип, например, BC547 или 2n2222 к выводам порта B.



Если применять динамическую индикацию с общим анодом, тогда понадобятся биполярные транзисторы p-n-p типа, например, BC557, эмиттеры

которых нужно подсоединить к плюсу «+» источника питания, а коллекторы – к минусу «-» также через подтягивающий резистор 10 кОм.



Алгоритм написания кода для подключения динамической индикации

Для большей конкретизации действий будем применять 4-х разрядный семисегментный индикатор с общим катодом. Первым делом следует создать массив цифр от 0 до 9. Далее необходимо разбить 4-хзначное число на четыре отдельных цифры. Например, число 1987 нужно разбить на 1, 9, 8 и 7. Затем единицу нужно отобразить в первом разряде индикатора, девятку – во втором, восьмерку – в третьем и семерку – в четвертом.

Среди многих алгоритмов разбивки многозначного числа на отдельные числа мы воспользуемся операциями деления и остатком от деления. Рассмотрим пример:

1987/1000 → 1

1987%1000/100 → 9

1987%100/10 → 8

1987%10 → 7

В языке С при использовании целочисленного типа данных ***int*** при выполнении деления все десятые, сотые и т. д., то есть все числа меньше

единицы отбрасываются. Остаются только целые числа. Математическое округление здесь не работает, то есть 1,9 в данном случае будет 1, а не 2.

Команда «остаток от деления» обозначается знаком процента «**%**». Данная команда отбрасывает все целые числа и оставляет остальную часть числа. Например, 1987%1000 → 987; 1987%100 → 87; 1987%10 → 7.

Далее следует написать команду, которая сначала отобразит первый разряд и соответствующее ему число, потом, через некоторый промежуток времени, второй разряд и отвечающее ему число; и так далее. Ниже приведен код с комментариями.

КОД

```
#define F_CPU 1000000L
#include <avr/io.h>
#include <util/delay.h>
#define CHISLO PORTD
#define RAZRIAD PORTB
unsigned int razr1 = 0, razr2 = 0, razr3 = 0, razr4 = 0;
unsigned int chisla [10] = {
    // числа от 0 до 10
    0x3f, 0x6, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x7, 0x7f, 0x6f
    };
void vse_chislo (unsigned int rabivka_chisla)
{
    razr1 = rabivka_chisla/1000;          // тысячи
    razr2 = rabivka_chisla%1000/100;  // сотни
    razr3 = rabivka_chisla%100/10;        // десятки
    razr4 = rabivka_chisla%10;            // единицы
}
int main(void)
{
  DDRB = 0b00001111;
  DDRD = 0b11111111;
  RAZRIAD = 0b00000001; // изначально 1-й разряд
  CHISLO = 0x3f;              // число 0
```

```
        while (1)
        {
            vse_chislo(1987);     // отображаемое число
                RAZRIAD = 0b00000001; // включаем 1-й разряд, остальные выключаем
            CHISLO = chisla [razr1]; // отображаем 1-ю цифру
            _delay_ms(3);
                RAZRIAD = 0b00000010; // включаем 2-й разряд, остальные выключаем
            CHISLO = chisla [razr2]; // отображаем 2-ю цифру
            _delay_ms(3);
                RAZRIAD = 0b00000100; // включаем 3-й разряд, остальные выключаем
            CHISLO = chisla [razr3]; // отображаем 3-ю цифру
            _delay_ms(3);
                RAZRIAD = 0b00001000; // включаем 4-й разряд, остальные выключаем
            CHISLO = chisla [razr4]; // отображаем 4-ю цифру
            _delay_ms(3);
        }
    }
```

Улучшаем программу для работы динамической индикации

Приведенный выше алгоритм является в большей степени обучающий, поэтому несколько упрощенный, но он также имеет место в программах, не выполняющих каких-либо быстрых расчетов в режиме реального времени. Единственным недостатком данного алгоритма является применение задержек, негативное влияние которых была рассмотрено ранее. Чтобы избавится от применения задержек можно применять прерывания от таймер-счетчиков. В ниже представленном коде задержки формируются с помощью нулевого таймер-счетчика и вызова прерывания по переполнению этого счетчика.

Для того, чтобы при каждом прерывании числа отображались последовательно в каждом разряде индикатора, добавлена переменная bc547, которая увеличивается на единицу при последующем вызове прерывания ISR

(TIMER0_OVF_vect). Затем выполняется проверка значения переменной bc547 и получает питания соответствующий разряд. Когда bc547 становится больше четырех, происходит сброс в единицу.

```
#define F_CPU 1000000L
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define CHISLO PORTD
#define RAZRIAD PORTB

unsigned int razr1 = 0, razr2 = 0, razr3 = 0, razr4 = 0;
unsigned char bc547 = 1;
unsigned int z = 0;

unsigned int chisla [10] = {
	// числа от 0 до 10
	0x3f, 0x6, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x7, 0x7f, 0x6f
};


ISR (TIMER0_OVF_vect) // вызов прерывания по переполнению 0-го таймер-счетчика
{

	if (bc547 == 1) {RAZRIAD = 0b00000001; CHISLO = chisla [razr1];}// зажигаем 1-ю цифру
	if (bc547 == 2) {RAZRIAD = 0b00000010; CHISLO = chisla [razr2];}// зажигаем 2-ю
	if (bc547 == 3) {RAZRIAD = 0b00000100; CHISLO = chisla [razr3];}// зажигаем 3-ю
	if (bc547 == 4) {RAZRIAD = 0b00001000; CHISLO = chisla [razr4];}// зажигаем 4-ю

	bc547++;
	if (bc547 > 4) bc547 = 1;

	}

void vse_chislo (unsigned int rabivka_chisla)
{
	razr1 = rabivka_chisla/1000;
	razr2 = rabivka_chisla%1000/100;
	razr3 = rabivka_chisla%100/10;
	razr4 = rabivka_chisla%10;
}

int main(void)
{
	DDRB = 0b00001111;
	DDRD = 0b11111111;

	RAZRIAD = 0b00000001; // изначально 1-й разряд
	CHISLO = 0x3f;				// число 0

	/* Настройка нулевого таймера на прерывание по переполнению*/

	TCCR0 |= (1<<1); TCCR0 &= ~((1<<0) | (1<<2)); // делим частоту на 8
	TCNT0 = 0; // В счетный регистр записываем 0
	TIMSK |= (1<<0); //Разрешаем прерывания по переполнению
	sei (); // Разрешаем глобальные прерывания

	DDRC &= ~((1<<4) | (1<<5));
	PORTC |= (1<<4) | (1<<5);

	while (1)
	{
		vse_chislo(z);		// отображаемое число
	}
}
```

Задание на выполнение:

1. Составить конспект по данной лекции.
2. Изучить принципы реализации динамической индикации.
3. Составить программу для циклического вывода на двухразрядный индикатор чисел от 0 до 99.

Результат программы отправить на e-mail: ***rasov@rambler.ru*** с темой **ДинамическаяИндикация_ФИО**