**AVR Урок 11. Динамическая индикация. Часть 1**

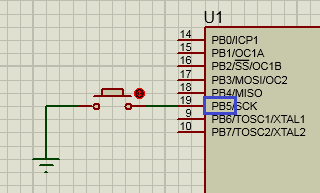
Сегодня мы попробуем подключить к микроконтроллеру не один одноразрядный индикатор, а два. То есть подключим ещё один такой индикатор. Только если мы подключим его точно так же, как и первый, то мы на это дело задействуем практически все ножки портов контроллера, что сделает наш проект нежизнеспособный.

Поэтому на помощь нам придет в этом случае другой способ — это **динамическая индикация**.

При динамической индикации в каждый момент времени мы будем выводить цифру только на одном индикаторе. То есть цифры, предназначенные для обоих индикаторов мы будем выводить по очереди. Но выводить по очереди мы их будем с такой частотой, что наш глаз этого не заметит. И нам будет казаться, что индикаторы работают постоянно, а не по очереди. Это позволяет нам подключить сегменты обоих индикаторов, а в последствии и большего количества индикаторов к одним и тем же ножкам порта, то есть параллельно. При таком способе будут сразу на оба индикатора подаваться по очереди сначала цифра, предназначенная для первого индикатора, а затем цифра, предназначенная для второго. И чтобы нам не виделись одновременно две цифры на каждом индикаторе, во время отображения цифры, предназначенной для определённого индикатора, мы будем на его анод положительное напряжение, а на другой анод не будем, а будем соединять его с общим проводом. Поэтому независимо от состояния его катодов светиться не будет ни один из его сегментов. Если на сегменте будет ноль, то вообще не будет разности потенциалов, а если будет единица, то р[**следующей части**](https://narodstream.ru/avr-urok-11-dinamicheskaya-indikaciya-chast-2/) азность потенциалов будет, но ток будет направлен в другую сторону и светодиод при таком направлении не будет светиться.

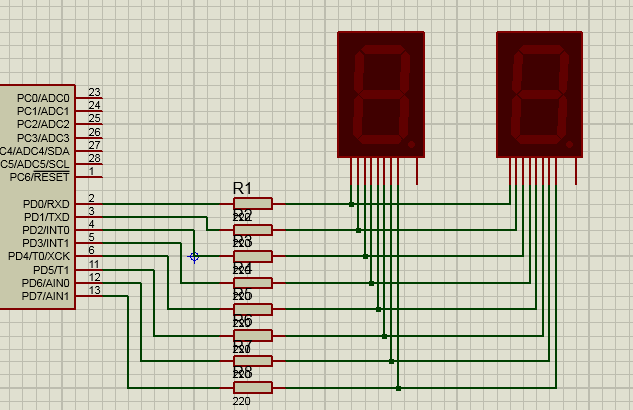
Новый проект мы создадим также как и обычно, скопировав код в main.c из одноименного файла прошлого проекта. Назовём проект **Test08**.

Давайте сначала соберём схему в протеусе, скопировав, как обычно, файл проекта из предыдущего занятия и подключив в с[**следующей части**](https://narodstream.ru/avr-urok-11-dinamicheskaya-indikaciya-chast-2/) войствах контроллера путь к новому проекту. Сначала мы переподключим кнопку на другую ножку порта. Зачем это нужно, мы увидим позже



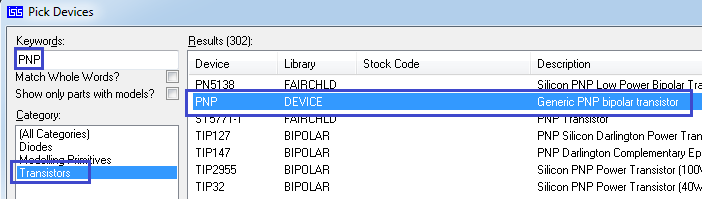
Отключим пока от питания общий анод и добавим ещё один такой же индикатор, какой у нас уже есть в проекте.

Для этого достаточно его просто выбрать в левой панели, заново из библиотеки его добавлять не нужно. Затем соединим катоды данного индикатора с одноимёнными катодами первого

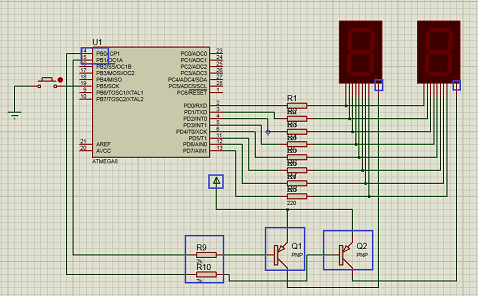


Освободившиеся аноды мы подключим к ножкам другого порта. Но подключим мы не напрямую, а через ключевые транзисторы, так как при свечении нескольких сегментов индикатора одновременно через ножку порта, подключенную к аноду индикатора, потечёт слишком большой ток. Порт может выйти из строя. Это нам вовсе не интересно. Поэтому подключим наши аноды к ножкам B0 и B1 через транзисторы. Вот зачем мы перенесли кнопку на другую ножку. Причём перенесли с запасом, так как подключать впоследствии мы будем четыре индикатора.

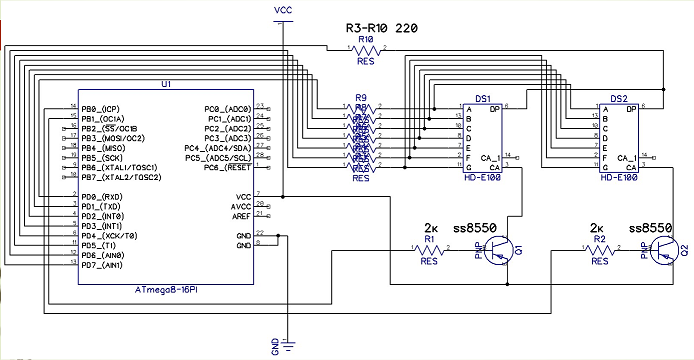
Добавим из библиотеки транзистор. Включать мы его будем в инверсном режиме, поэтому выбираем структуру p-n-p



Подключим таких два транзистора в схему, добавив ещё два резистора по 2 килоома для ограничения тока базы. Эмиттеры подключим к питанию, а коллекторы — к анодам индикаторов (нажмите на картинку для увеличения изображения)

[](https://narodstream.ru/wp-content/uploads/2016/11/Image03_0955.png)

Также я покажу полную схему нашей сборки (нажмите на картинку для увеличения изображения)

[](https://narodstream.ru/wp-content/uploads/2016/11/Image04_1387.png)

Здесь мы видим, что транзисторы применены здесь ss8550. Стоят они копейки, но в качестве ключей работают прекрасно.

Со схемой пока всё.

А проектом мы займёмся в [**следующей части**](https://narodstream.ru/avr-urok-11-dinamicheskaya-indikaciya-chast-2/) нашего занятия

**Урок 11**

**Часть 2**

# ****Динамическая индикация****

В [прошлой части](https://narodstream.ru/avr-urok-11-dinamicheskaya-indikaciya-chast-1/) нашего занятия мы познакомились с принципом динамической индикации, собрали схему в протеусе.

Теперь закончим со сборкой и перейдём в проект, так как без соответствующего кода ничего работать на будет.

Начнем с порта B. Он у нас был сконфигурирован весь полностью на вход, но теперь так дело не пойдёт. Давайте половину порта сделаем на вход, а ту половину, где у нас подключены через транзисторы аноды индикаторов, сконфигурируем на выход. А подтягивающий резистор на ножке порта B переместим на другую ножку, к которой мы переподключили кнопку

DDRD = 0xFF;

DDRB = **0b00001111**;

PORTB = 0b00**1**0000**0**;

Теперь, если мы соберём код и запустим его в протеусе, то у нас будут работать одновременно два индикатора. Это нам не подходит. Поэтому думаем дальше.

Прибавим разрядность переменной **i**. Пока под две цифры нам и char подойдёт, но на будущее нам этого не хватит. Поэтому сделаем это заранее

//———————————————

**unsigned int i;**

И ещё нам под цифры каждого разряда потребуются также переменные

unsigned int i;

**unsigned char R1=0, R2=0;**

//———————————————

Также добавим ещё одну функцию, которая будет заниматься выводом двухзначной цифры на дисплей. Поэтому в качестве входного параметра здесь будет **unsigned int**. Добавим данную функцию перед функцией main()

//———————————————

**void ledprint(unsigned int number)**

**{**

**}**

//———————————————

В данной функции мы распределим всю двухзначную цифру по разрядам.

Для этого мы применим математическую операцию, которая вычисляет остаток от деления. Обозначается она знаком **%**. Через данную операцию мы вычислим единицы. А затем, чтобы вычислить десятки, мы просто поделим на 10 входной параметр. Но так как у нас везде целочисленные операнды, то в результате получится целое число с выброшенным остатком, что и будет соответствовать десяткам, так как число у нас двухзначное

void ledprint(unsigned int number)

{

**R1 = number%10;**

**R2 = number/10;**

}

А таймер мы сегодня применим не для того, чтобы он наращивал цифры, а для того, чтобы он поочерёдно зажигал цифры на индикаторах. Добавим ещё одну переменную. Добавить мы её также можем необязательно в начале файла, а можем непосредственно перед кодом функции-обработчика прерываний, главное чтобы переменная в функции уже была определена

//———————————————

**unsigned char n\_count=0;**

//———————————————

ISR (TIMER1\_COMPA\_vect)

И в зависимости от значения данной переменной, мы будем включать определённую ножку порта B, разрешая при этом вывод цифры на соответствующем индикаторе. Другую ножку порта мы будем отключать. Таже в том же условии мы будем вызывать функцию, которая будет включать комбинацию сегментов в зависимости от посланной переменной. Но так как определённая ножка порта B будет у нас отключена, то сегменты другого индикатора светиться на будут. Затем после всего этого мы будем прибавлять на 1 (инкрементировать) данную переменную. Но делать мы это будем до тех пор, пока она не достигнет значения 1. Вообщем, в связи с этим, переменная n\_count будет равна либо 1, либо 0, так как пока у нас только 2 индикатора и соответственно будет только 2 варианта

ISR (TIMER1\_COMPA\_vect)

{

**if(n\_count==0) {PORTB&=~(1<<PORTB0);PORTB|=(1<<PORTB1);segchar(R1);}**

**if(n\_count==1) {PORTB&=~(1<<PORTB1);PORTB|=(1<<PORTB0);segchar(R2);}**

**n\_count++;**

**if (n\_count>1) n\_count=0;**

}

Теперь нам осталось только лишь в функцию ledprint() отправить какую-нибудь цифру. Давайте пока подадим любую и наращивать её пока не будем. Для этого мы вызовем функцию в main() после всех инициализаций, отправив в неё, например, число 97

sei();

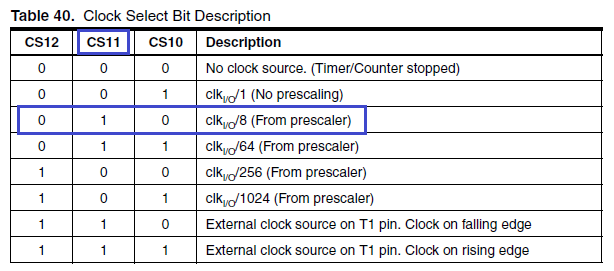
**ledprint(97);**

while(1)

Попробуем собрать проект и запустить его в протеусе. Мы видим, что код работает, динамическая индикация присутствует, но слишком наглядно, так как цифры светятся по очереди раз в 2 секунды.

Для наглядности работы динамической индикации это хорошо, но для постоянной работы не пойдёт. Глаз вообще не должен видимть динамику индикации. Должно быть впечатление, что цифры светятся постоянно.

Поэтому нам нужно настроить таймер на другую частоту. Попробуем порегулировать частоту делителем. Сделаем делитель 8 вместо 256 и, произведя нехитрые расчеты мы получим частоту 32. но так как у нас индикатора два, то получится на каждый по 16 герц. Этого конечно маловато, но попробовать можно. Чтобы нам выставить делитель 8, то мы в регистре **TCCR1B** должны включить бит **CS11**



Так и поступим

  OCR1AL = 0b01000010;

 TCCR1B |= (1<<**CS11**);//установим делитель

}

Соберём код и посмотрим работу в протеусе. Вроде бы не мерцает. На живых индикаторах пока смотреть не будем. Если что, то потом подравим частоту. Пока займёмся счетчиком, нам нужно будет считать до 99, а затем начинать с нуля. Для этого раскомментируем весь код в бесконечном цикле, также раскомментируем переменные для кнопки

int main(void)

{

**unsigned char butcount=0, butstate=0;**

Также, раз уж считать мы будем до 99, то также исправим код вот здесь

while(1)

{

  for(i=0;i<**100**;i++)

  {

То есть мы попробуем также кнопкой обнулять счётчик. Но так как мы кнопку переключили на другой пин, то нужно немного подправить проверяемое значение

if (!(PINB&0b**00100000**))

Также вместо функции **segchar** будет функция **ledprint**

**ledprint(i);**

\_delay\_ms(500);

Ну, собственно, и всё.

Соберём наш код и проверим его сначала в протеусе. Всё у нас считается и сбрасывается кнопкой. Отлично!

Теперь прошьём контроллер и посмотрим на живых индикаторах работу кода.

Код работает, но индикаторы мерцают. Поэтому немножко ещё поиграем с частотой в таймере. Если убрать делитель совсем, то эксперементальным путём было выявлено, что наши индикаторы перестанут светиться вообще. Этот эксперимент вы можете увидеть во второй части видеоурока (ссылки на видеоурок внизу, достаточно кликнуть по нужной картинке, в принципе, как и в любом уроке). Поэтому придется нам поиграть с цифрой в регистре OCR1A. Причём мы докажем, что виновата не слишком высокая частота, а то, что без делителя просто отказывается работать таймер. Поэтому цифру мы рассчитаем так, чтобы частота при делителе была примерно такая же как и без делителя, но на предыдущем значении в OCR1A. Цифру мы получим примерно 3906. Превратим её в двоичный вид и занесём в регистровую пару OCR1A. Код станет вот таким

**OCR1AH = 0b00001111; //записываем в регистр число для сравнения**

**OCR1AL = 0b01000010;**

Теперь, если собрать код и прошить контроллер, мы увидим, что всё работает как надо

