**AVR Урок 4. Смотрим результаты работы**

Posted on [Ноябрь 3, 2016](https://narodstream.ru/avr-urok-4-smotrim-rezultaty-raboty/) by [https://secure.gravatar.com/avatar/4824b24065500834db4b9f331b608833?s=32&d=mm&r=gNarod Stream](https://narodstream.ru/author/admin/) Опубликовано в [GPIO](https://narodstream.ru/gpio/), [Программирование AVR](https://narodstream.ru/rub_avr/) — [22 комментария ↓](https://narodstream.ru/avr-urok-4-smotrim-rezultaty-raboty/" \l "comments)

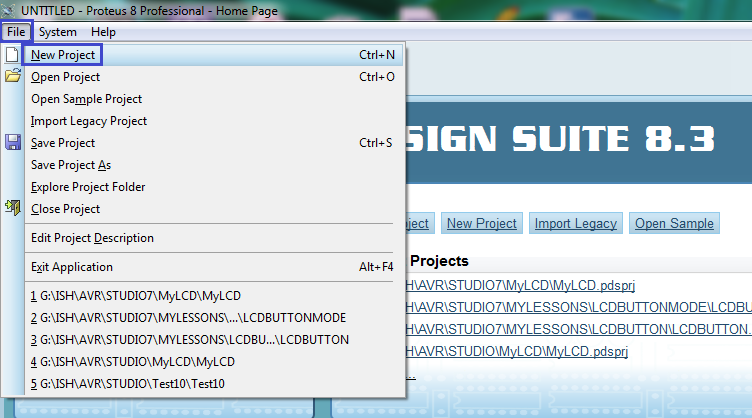
**Урок 4**

**Смотрим результаты работы**

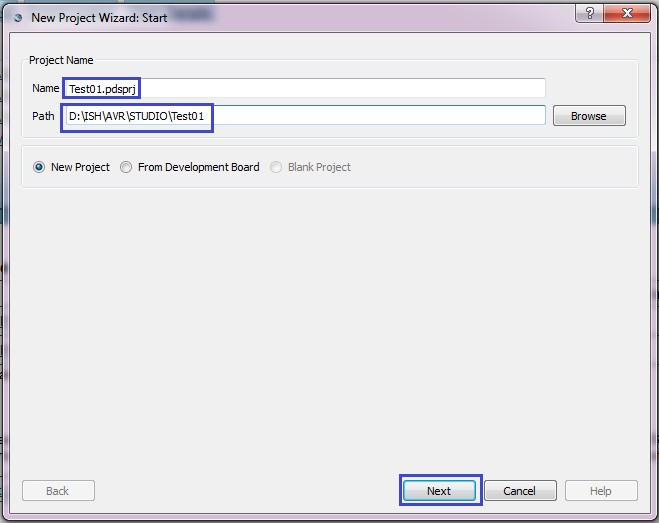
Сегодня мы продолжим начатое дело и попытаемся увидеть результаты работы нашего кода в живом отладчике — в **протеусе**. Здесь уже симуляция происходит наглядно. Вот его ярлык на рабочем столе



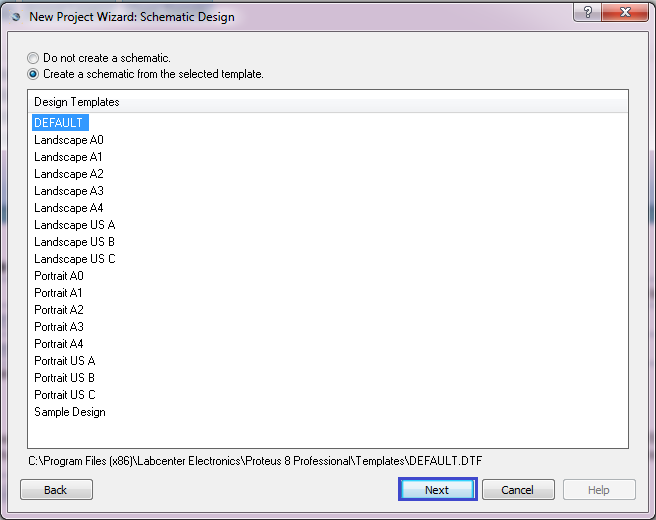
Где найти данную программу, я рассказыать не буду, вы сами её найдёте без труда. Расскажу, как ею пользоваться. Запустим его и создадим новый проект



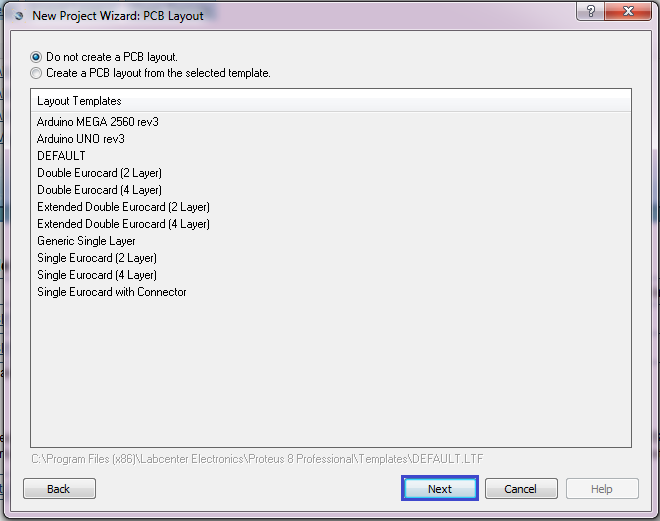
Имя проекта у нас будет такое же, как имя проекта в Atmel Studio — **Test01**и создадим мы его в папке нашего проекта

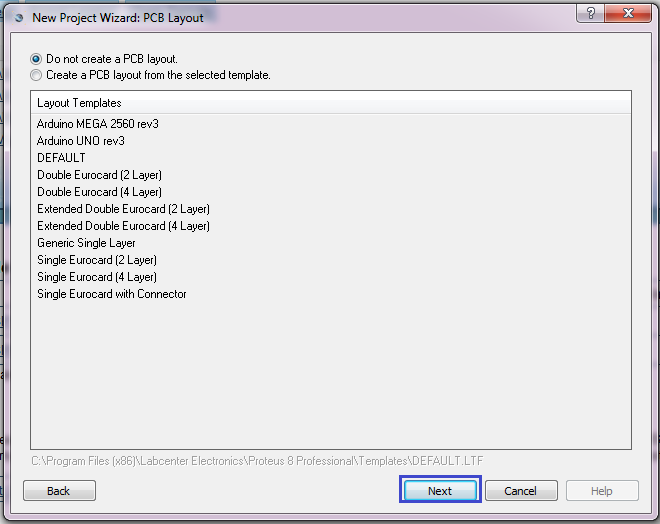


Идем дальше, всё везде оставляем по умолчанию

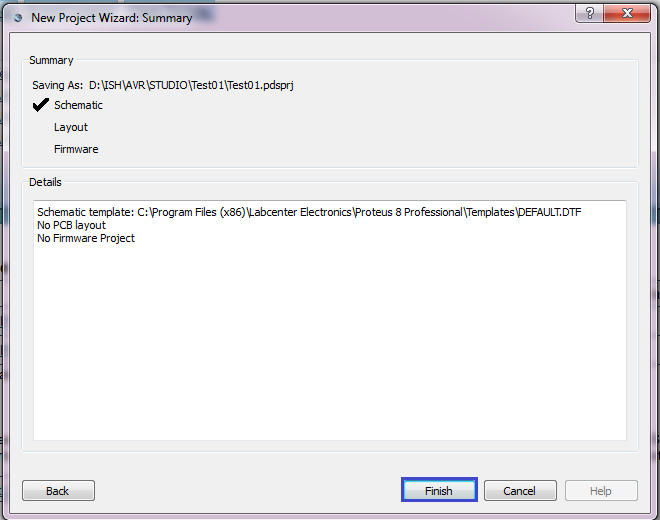


Далее всё по умолчанию



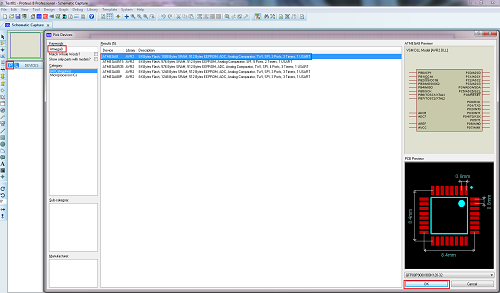


В конце жмём "Finish"

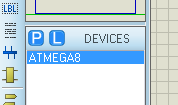


Проект создан.

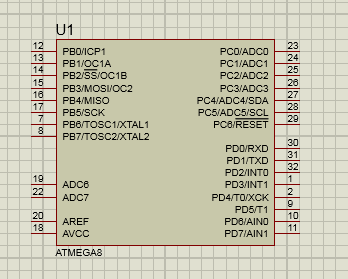
Давайте добавим в него какой-нибудь элемент. Для этого нажмём на кнопку "P" в левой панели и из списка выберем элемент. Сверху в строке поиска можно вводить наименование или часть наименования элемента. Попробуем добавить наш микроконтроллер Atmega8. Здесь контроллеры без букв. Выберем именно Atmega8 и нажмем "ОК" либо приеним двойной щелчок по строке с элементом (нажмите на картинку для увеличения размера)

[](https://narodstream.ru/wp-content/uploads/2016/11/009_1675.png)

Элемент появится в левой панели

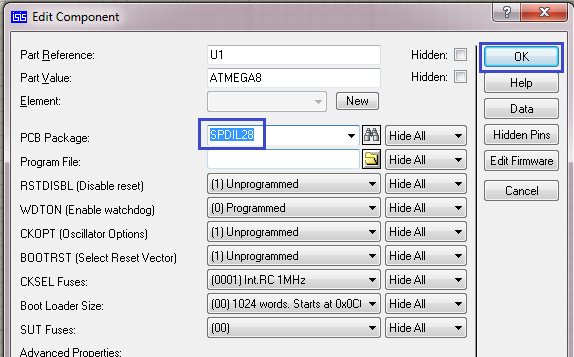


Если мы только что добавили элемент, то, чтобы его установить в рабочем пространстве, выбирать его не нужно, достаточно щёлкнуть по тому месту рабочего пространства, куда мы хотели бы установить данный элемент. По первому щелчку элемент установится, но ещё не "прилипнет" к пространству, то есть мы ещё можем его позиционировать. После второго щелчка наш контроллер жёстко установится.

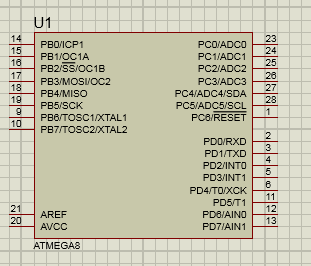


Мы вижим с вами, что это не совсем тот тип контроллера, который мы будем использовать. Нам нужен вариант в корпусе DIP с 28 ножками. Чтобы изменить тип корпуса, щёлкнем двойным щелчком по контроллеру и изменим его тип в следующем выпадающем списке. Также заодно в поле "Program File" мы добавим файл с прошивкой с помощью кнопочки с папкой справа от поля. В качестве прошивки у нас будет файл, созданный на прошлом занятии с помощью Atmel Studio. Путь к нему вы видите в поле на рисунке, у вас, соответственно, он может отличаться

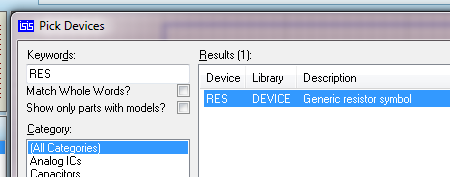
и нажмем "ОК"

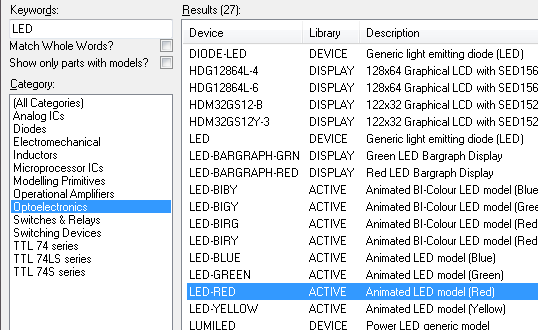


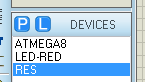
Теперь мы видим, что это именно тот контроллер

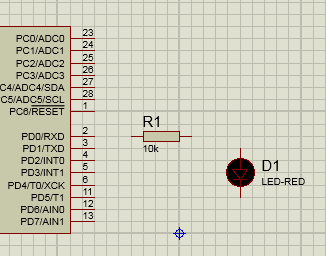


Приблизительно тем же способом мы добавим резистор и светодиод на площадку. Резистор нужен для того, чтобы ограничить ток, протекающий через светодиод для того, чтобы не сгорели ни светодиод, ни контроллер.

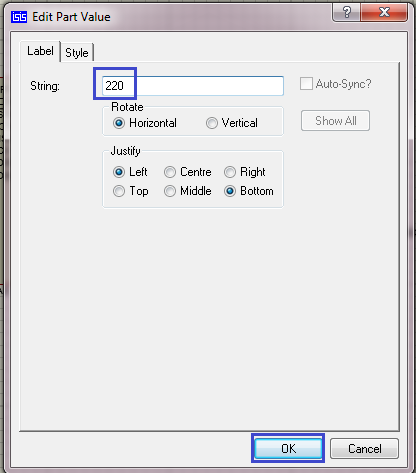




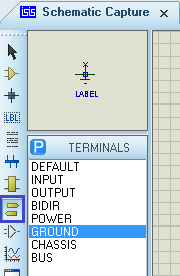


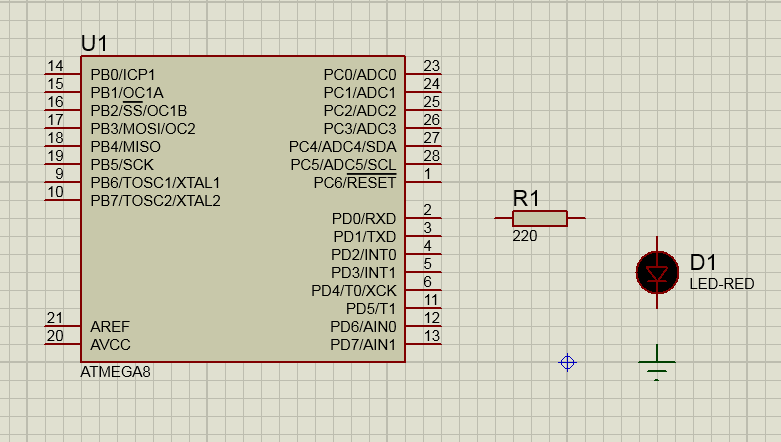


Только для токоограничения светодиода 10 килоом будет слишком много. Чтобы изменить номинал, щёлкнем двойным щелчком по надписи "10к" либо по самому резистору и заменим значение на 220 ом. При 5 вольтах это будет оптимально

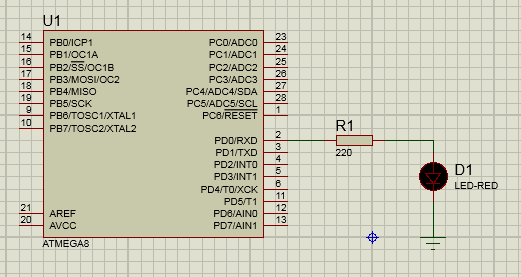


Ещё нам нужно будет добавить контакт для общего провода (массу). Для этого нажмём в левой панели определённый значок и выберем из списка "GROUND". Затем таким же образом, как и любые элементы, добавим его в нужное место





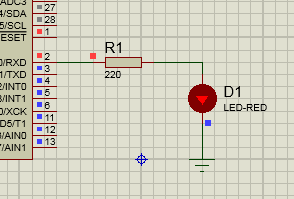
Ну вот наши элементы на месте. Осталось их соединиить. Делается это с помощью мыши. Сначала мы щёлкаем одиночным щелчком по одному контакту соединения, а затем по другому. Таким образом мы соединим все элементы



Наши элементы соединены, можно попробовать схему в работе. Питание подводить к контроллеру необязательно. Это подразумевается само собой. Для запуска в нижней панели нажмём кнопочку в виде зелёного треугольника. Кнопочка станет ярко-зелёной

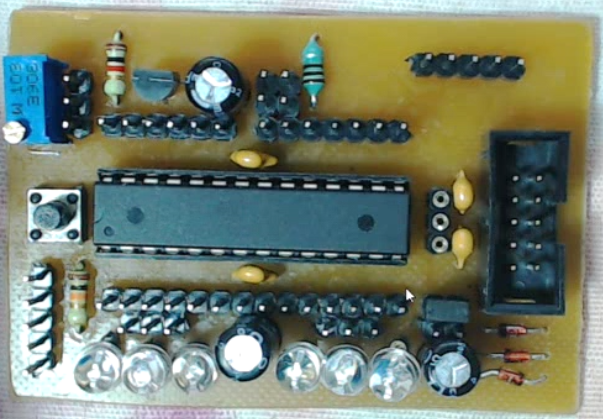
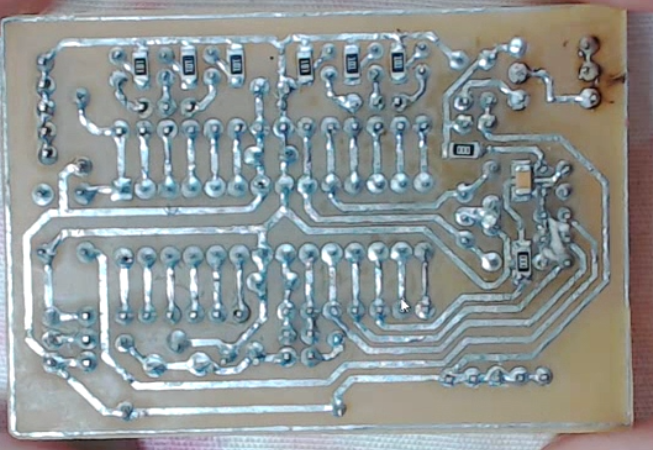
023

Если сделано всё правильно, то светодиод будет светиться



Таким образом мы увидели результат того, чего мы добились в результате кода. Наша программа работает. Чтобы остановить процесс, нажимаем на кнопочку с синим квадратом.

Следующая задача урока — посмотреть результат работы программы в железе — на живом контроллере. Для этого воспользуемся отладочной платой, которая есть у меня.

Данная плата была взята [здесь на этом сайте](http://cxem.net/mc/mc292.php).

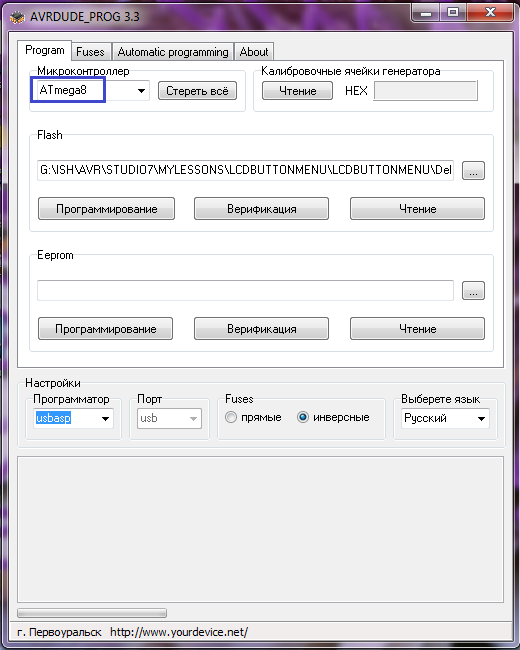
Подключать к компьютеру эту отладочную плату мы будем вот таким программатором



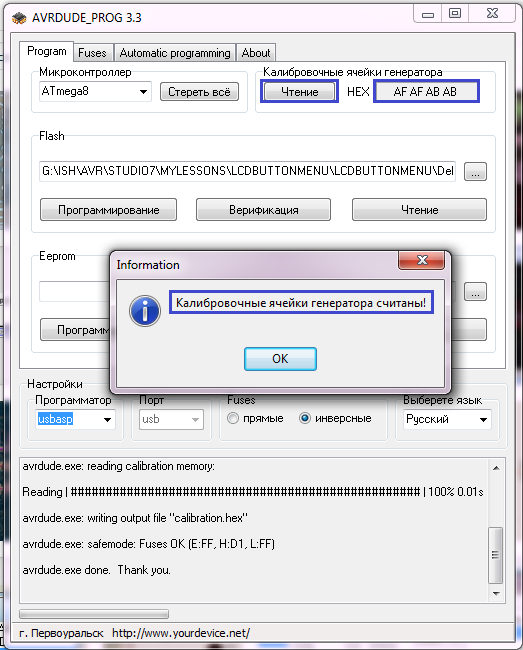
На данный программатор у меня есть [обзор](https://www.youtube.com/watch?v=p-F_bn2mjO4). Там есть информация, как данный программатор подсоединить, как установить на него драйвер и как с помощью него прошивать контроллеры. В данном обзоре мы прошиваем три различных контроллера. Также используем мы несколько программ для прошивки. В данном уроке мы будем использовать программу avrdude33.

Подключим отладочную плату к программатору, а программатор подключим к шине USB компьютера. Драйвер должен быть установлен.

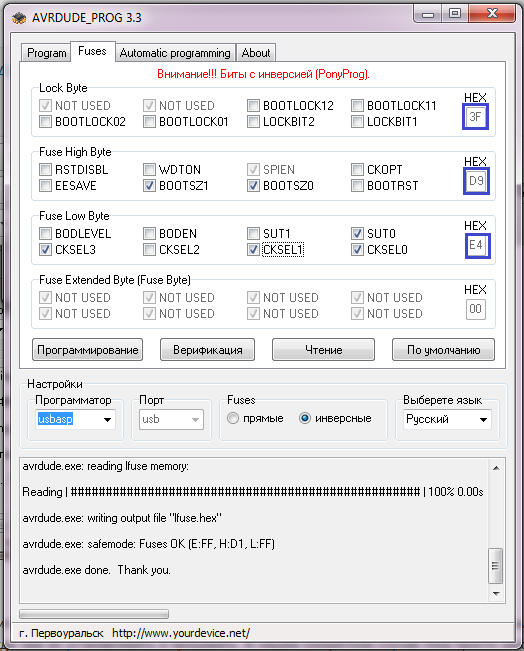
Теперь Запустим прошивочную программу и выберем там наш контроллер



Попытаемся считать конфигурационные ячейки контроллера, нажав для этого кнопку "Чтение". Если всё подключено и установлено правильно, то мы должны увидеть вот такое вот сообщение



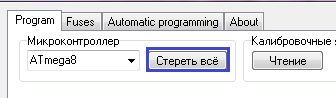
Дальше открываем закладку Fuses. Фьюзы — это настроечные биты контроллера, эксперементировать с которыми — занятие опасное. Поэтому внимательно изучите техническую документацию на тот контроллер, который программируете, прежде чем их устанавливать и прошивать. В нашем конкретном случае пока просто запомните установки на рисунке и обязательно выставите точно такие же, если конечно вы используете такой же контроллер и он у вас также сконфигурирован, то есть у вас нет кварцевого резонатора и вы пользуетесь именно встроенным генератором тактовой частоты. Фьюзы должны быть именно "инверсные", поэтому кнопочка должна быть установлена именно возле данной опции



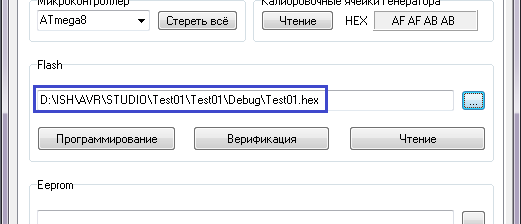
Данные настройки фьюзов заставят работать контроллер Atmega8A от внутреннего генератора тактовых импульсов с частотой 8 мегагерц. Чтобы сохранить настройки в контроллер, нажимаем "Программироване", ещё несколько раз перед этим хорошенько подумав, иначе можно получить кирпич.

Микроконтроллер Atmega8a может с равным успехом работать на частоте 1, 2, 4, 8 мегагерц от внутреннего генератора, также он может работать от внешнего генератора, а также с использованием кварцевого резонатора. В последнем случае обеспечивается более стабильная работа внутреннего тактового генератора и также контроллер уже сможет работать ещё и на частоте 16 мегагерц.

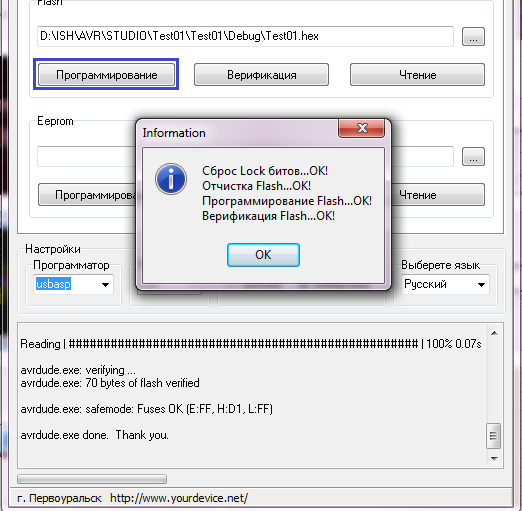
Теперь вернёмся в закладку "Program". Также перед тем, как заливать нашу прошивку, я бы советовал вам стирать информацию в контроллере нажатием кнопки "Стереть всё". Поэтому нажимаем эту кнопочку



Теперь выбираем нашу прошивку — именно ту, которую мы испытывали в протеусе



Теперь нажимаем "Программирование"



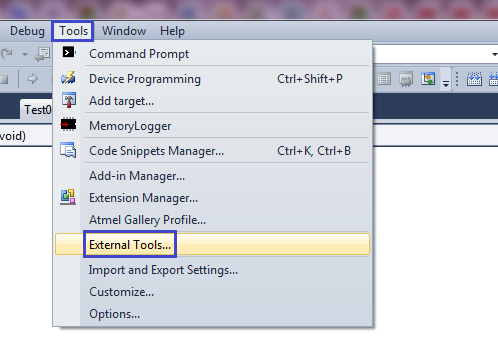
Наша прошивка залита в контроллер. Перезагрузка происходит автоматически.

Как мы видим, наша программа работает и светодиод светится



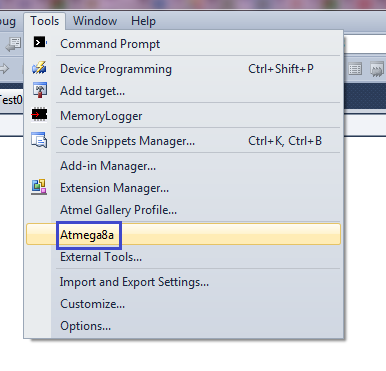
Питается данная плата от напряжения 5 вольт, идущего с программатора, программатор данное напряжение берёт от порта USB. То есть отдельно на данную плату при условии небольшой нагрузки на порты питание подводить на требуется, что очень удобно для учебных целей.

Ну и ещё давайте попробуем добиться программирования нашего контроллера не через постороннюю программу, а напрямик из Atmel Studio. Вообще я предпочитаю программировать именно из Avrdude, но вообще напрямую удобнее, если мы не меняем фьюзы и не выбираем какой-то другой контроллер. Это именно такой случай, когда наш программатор не имеет поддержки драйверной в Atmel Studio. Зайдём в студию, откроем наш проект и зайдём в меню Tools -> External Tools



Назовём наше новое меню "Atmega8a". В строку "Command" через обзорную кнопку выбираем файл с программой Avrdude, только не тот файл, которым мы запускаем данную программу, вернее её графическую оболочку, а тот файл, который используется непосредственно для команд командной строки. В следующей строке "Arguments" вписываем строку с параметрами "-c usbasp -p m8 -B12 -U flash:w:$(ProjectDir)Debug\$(ItemFileName).hex:a" только без кавычек. Жмем "Apply" и "OK".

На всякий случай ещё раз соберём наш проект и попробуем с помощью нашей команды прошить контроллер, воспользовавшить появившемся новым пунктом меню Atmega8a в меню Tools.



В результате должен загореться светодиод. Такой вариант намного удобнее, но только он не позволяет редактировать фьюзы (оно и не часто требуется) и также не очищает флеш-память контроллета (но без этого также всё нормально прошивается и работает).

[*Предыдущий урок*](https://narodstream.ru/avr-urok-3-pishem-kod-na-si-zazhigaem-svetodi/) [*Программирование МК AVR*](https://narodstream.ru/programmirovanie-mk-avr/) [*Следующий урок*](https://narodstream.ru/avr-urok-5-migayushhij-svetodiod/)