

Инструкция по использованию встроенного JTAG отладчика

В демоплату Pinboard встроен мощный отладочный инструмент, позволяющий отлаживать программу пошагово, прямо в реальном железе. Причем вне зависимости от языка программирования – Си или Ассемблер, главное чтобы была отладочная информация подготовлена. Если работа ведется в AVRStudio то вообще ничего не надо делать – достаточно только запустить режим отладки.

Что нужно сделать для того, чтобы воспользоваться JTAG отладчиком.

1. Подключить линию отладки JTAG:

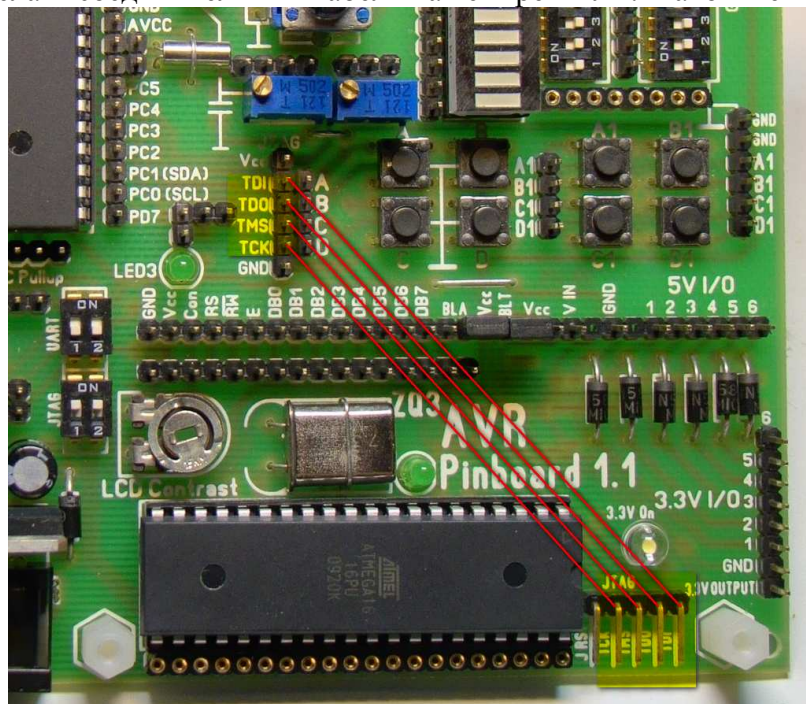
Для этого нужно соединить JTAG адаптер с выводами JTAG интерфейса главного контроллера. Это выводы **TDI, TDO, TMS, TCK**

Соединять нужно один к одному. Т.е.

- ▶ **TDI – TDI**
- ▶ **TDO – TDO**
- ▶ **TMS – TMS**
- ▶ **TCK – TCK**

На вывод **J RST** со стороны JTAG адаптера не обращайте внимания, он для прошивки JTAG контроллера.

Для этого придется сделать соединительный кабель на четыре жилы. Такой же как для прошивки.

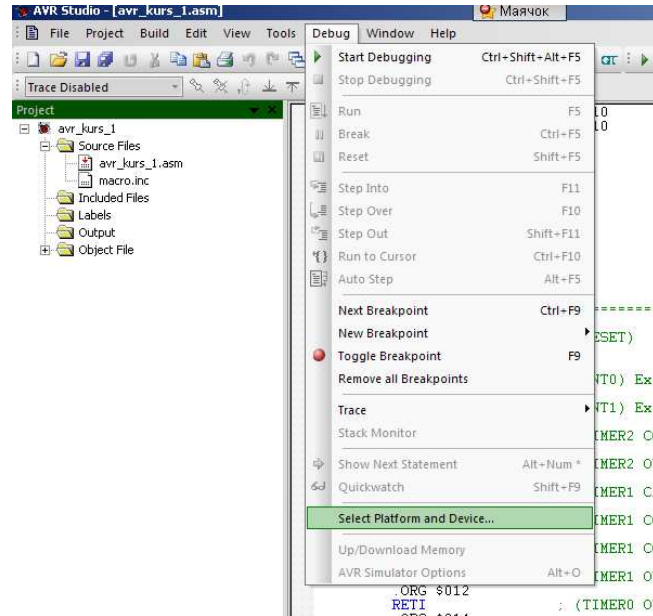


2. Подключить JTAG контроллер к интерфейсу USB-UART, переключив рубильники UART в положение OFF, а рубильники JTAG в положение ON

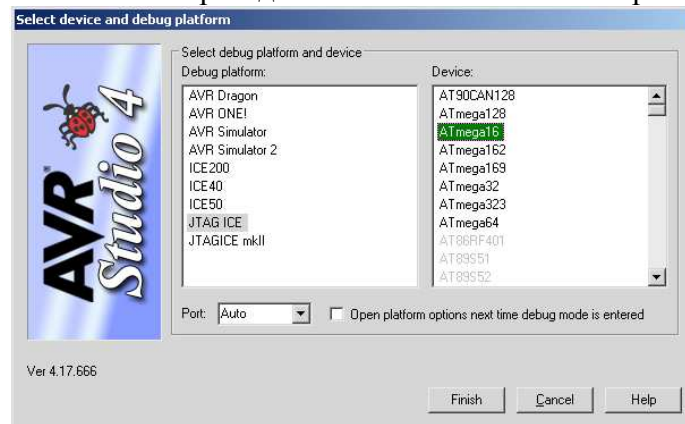


- Убедиться, что порт, на который села микросхема FT232RL соответствует COM1...COM4 иначе студия не найдет отладочный интерфейс. Если это не так, то смотри инструкцию по быстрому старту демоплаты, там подробно указано как и что нужно подправить.
- Настроить студию так, чтобы в качестве отладочного инструмента был не AVR Simulator, а JTAG ICE (не путать с JTAG ICE II)

Делается это в меню DEBUG->Select Platform & Device

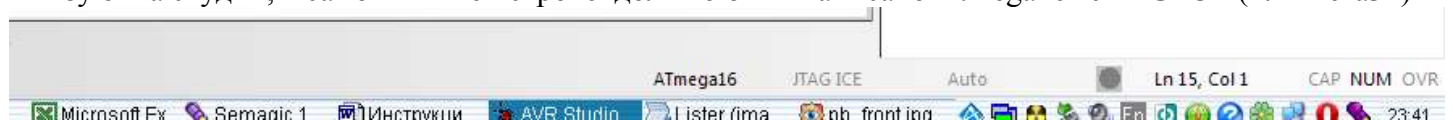


Там заново выбрать контроллер и отладочную платформу (либо при создании нового проекта сразу указать, что будет JTAG), не забыть также повторно выбрать тип процессора. В данном случае будет Mega16 или Mega32 (в зависимости от серии демоплаты и главного контроллера):



Жметса Финиш и студия готова к работе.

Внизу окна студии, в самой нижней строке должно быть написано Atmega16 JTAG ICE (или мега32)



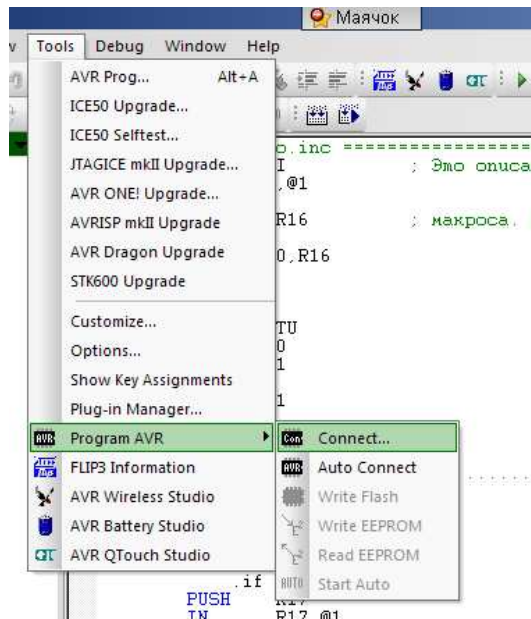
!!!WARNING!!!

При подключении JTAG интерфейса в режиме отладки происходит перезапись памяти кристалла. А это значит из контроллера будет стерт Bootloader. Учитывайте это. Залить бутлоадер можно тем же JTAG адаптером, используя его как программатор. Либо через FTBB Prog, воспользовавшись соответствующей инструкцией.

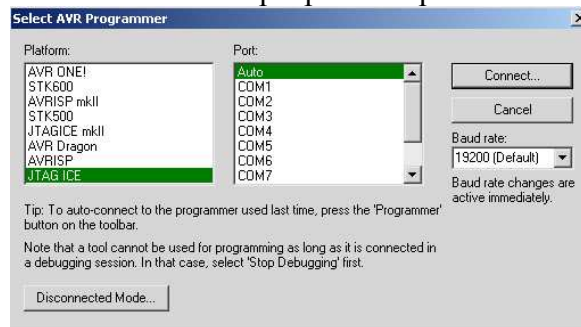
!!!WARNING!!!

При работе в режиме отладки через JTAG fuse биты не меняются, а значит, не смотря на стертый бутлоадер, контроллер будет стартовать словно с бутом – т.е. не с нулевого адреса, а с адреса старта бутлоадера. В этом случае нужно будет подправить FUSE биты, переключив FUSE бит BOOTRST в противоположное состояние. Сделать это можно тем же JTAG адаптером.

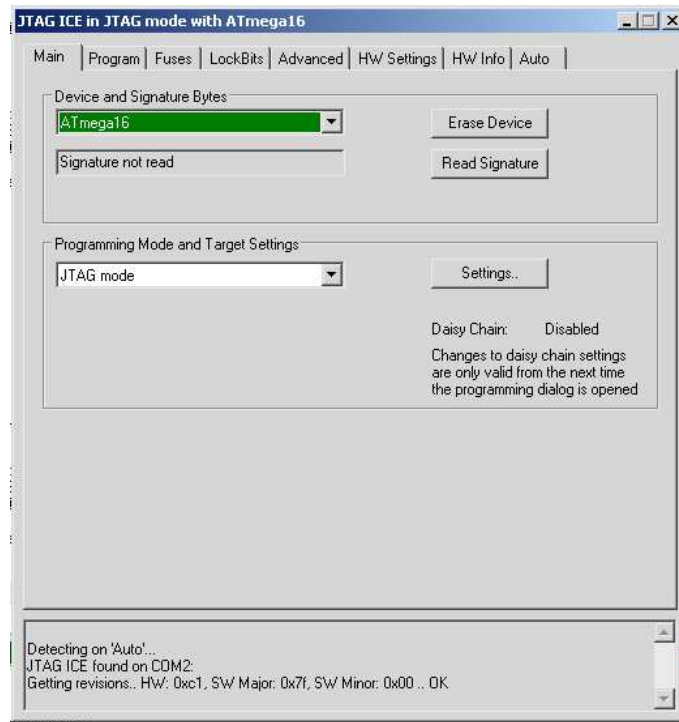
Вызовите диалог **Connect...**



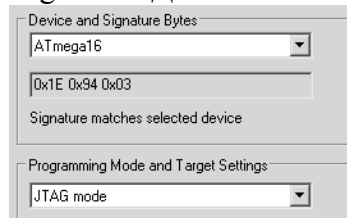
В типе коннекта надо выбрать JTAG в качестве программатора.



И нажать Connect... Адаптер немного потупит и выдаст окно программатора:

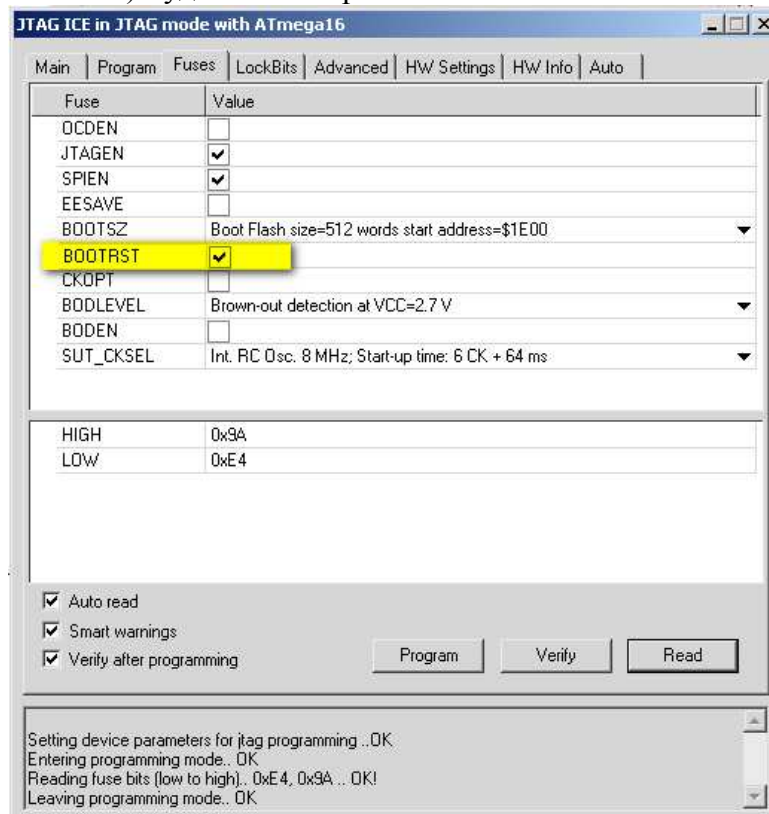


Выбери тип контроллера и нажми Read Signature. Должна считаться сигнатура и появиться надпись:



Это говорит о том, что сигнатура соответствует текущему чипу. Это очень важно.

Далее переходим на вкладку Fuses и жмем кнопку Read (всегда при работе с Fuses нужно считывать их перед тем как что либо менять!!!) Будет такая картина:



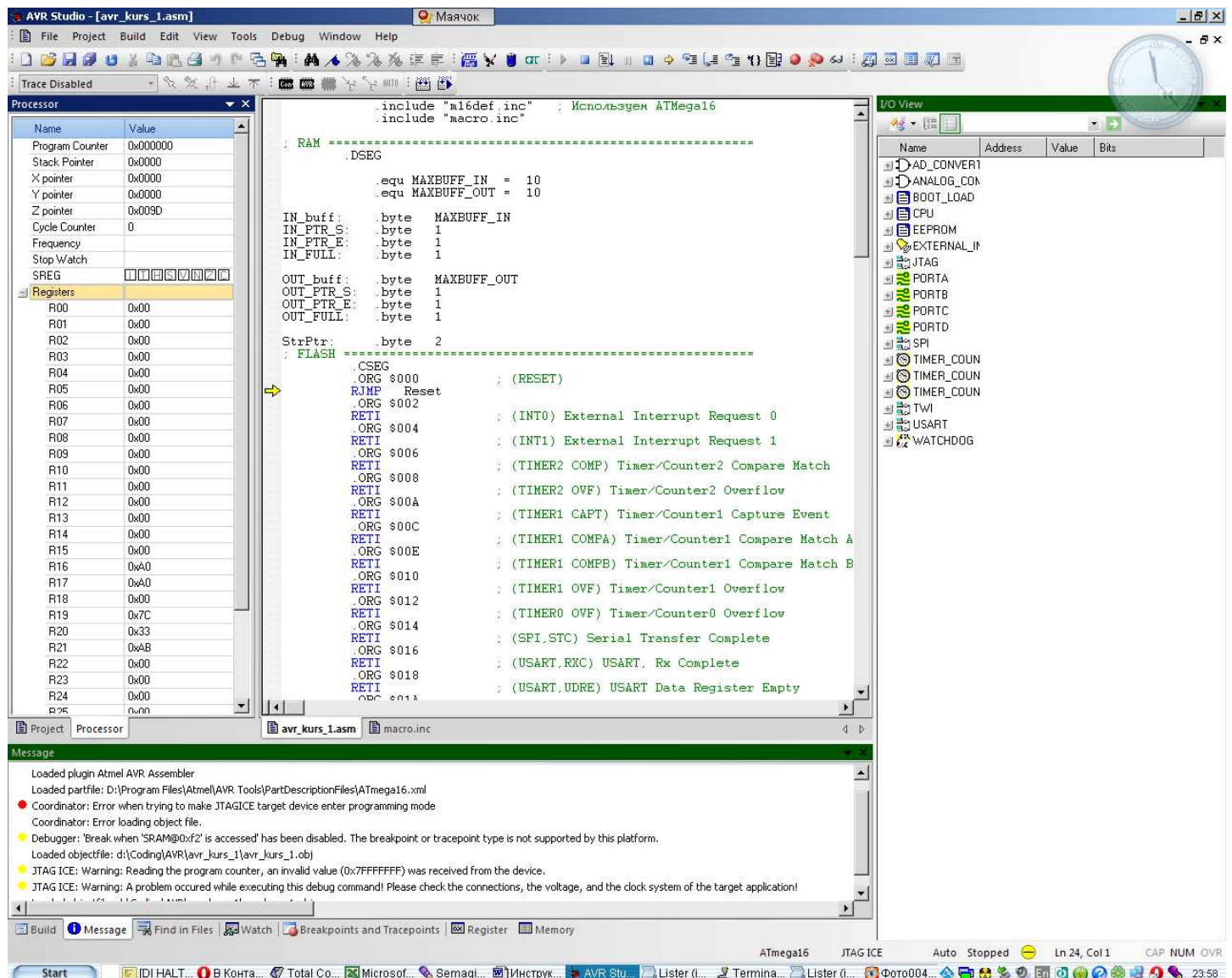
Галочку BOOTRST надо снять, после чего нажать Program. И закрыть окно прошивальщика.

Кстати, этим прошивальщиком можно заливать и любую произвольную прошивку, достаточно заглянуть на вкладку Program. А также вернуть на место Bootloader и выставить любые Fuse биты.

Все, теперь мы готовы к запуску. Жми Compile and Run. Как обычно:

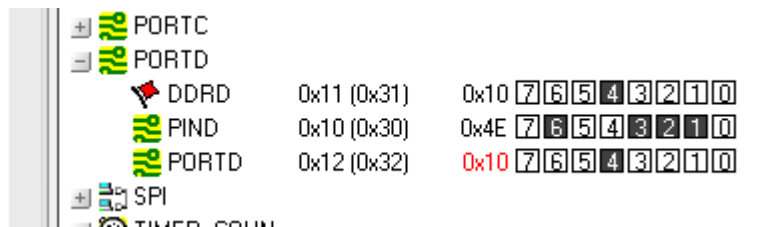


Пойдет компиляция, потом студия затупит, а JTAG адаптер начнет активно моргать лампочкой. Спустя несколько секунд. Стрелочка текущей команды появится возле метки ORG 0000 – программа запущена и готова к пошаговому выполнению.



Форма студии очень гибко настраивается, поэтому расположение полей может сильно отличаться.

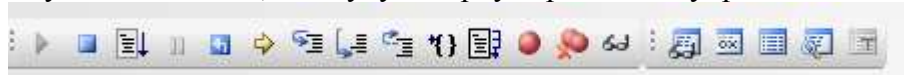
Стрелка показывает текущую инструкцию, поле IO View – содержимое регистров периферии. Причем их можно менять произвольно, вне зависимости от программы. Разверните PORTD, выставите галочки в биты DDRD4 и PORTD4




















И при этом светодиод LED1, если у него будет стоять джампер, зажгется! А ведь процессор еще ни одного шага не сделал! Он по прежнему стоит на нуле, но мы можем дрыгать периферию как нам вздумается. Если потыкать сейчас по регистрам таймеров, то можно запустить ШИМ, а можно сгенерировать прерывание. И все это в реальном контроллере, не написав ни единой строчки кода, только вручную переключив битами регистров. При этом в регистре PIN будет показано текущее значение, которое навелось на портах контроллера на момент обмена.

Работа в режиме отладки

Изучите кнопки, что будут вверху экрана. Они управляют выполнением кода



Слева направо:

-  Пуск (сейчас запущена, поэтому серая)
-  Стоп – остановка симуляции (защита при симуляции программа останется в МК)
-  Свободный пуск – контроллер пойдет выполнять программу, не считываясь о результатах.
-  Пауза. Позволяет выбросить выполняющуюся программу в том месте где мы нажали.
-  Сброс контроллера – словно бы мы нажали RESET. Удобней сбрасывать так, чем перегружать симуляцию.
-  Перейти к текущему положению программы. Удобно когда у нас прога раскидана по куче файлов, чтобы не искать везде эту желтую стрелочку
-  Шаг со входом в процедуры/функции. Фактически пошаговое выполнение.
-  Шаг с перескакиванием функций. Удобно когда надо перескочить уже отлаженные и четко работающие функции, дабы не терять на них время.
-  Шаг над. При этом программа делает очередную итерацию главного цикла и встает над точкой со стрелкой. Если конечно сможет. Зависит от алгоритма программы.
-  Выполнить код до текущего положения курсора. Адски полезная фишка
-  Пошаговое автовыполнение. Студия сама будет быстро делать шаги. Бесплезная опция =)
-  Включить брейпоинт. Точка останова, на которой контроллер встанет как вкопанный. Очень удобная вещь. Незаменима при отладке. Ставится в нужном месте, где нам надо посмотреть выполнение кода и делается свободный запуск программы. Когда студия пойдет через бряк она там вывалится с показом всей инфы.
-  Добавить переменную в Watch List. Ставим курсор на любую переменную и ждем эту кнопку. И она попадает под наше пристальное наблюдение.
-  Показать/Спрятать окно гляделок за переменными.
-  Показать/Спрятать Окно регистров.
-  Показать/Спрятать окно памяти. Там можно смотреть как движется стек, как расположены данные в ПЗУ, ОЗУ, ЕЕПРОМЕ. Полезнейшая вещь!
-  Переключиться в режим дизассемблера. Даже когда пишешь на асме приговждается, например, когда надо развернуть макрос и обработать его не как одну команду, а покомандно. А уж когда пишешь на Си без дизасма никуда. Иначе непонятно что вообще

там происходит. Оптимизатор бывает такое отмочит, что без курения в машинные коды и не поймешь что происходит.

Подробнее про отладку в студии читайте в статье:

<http://easyelectronics.ru/avr-uchebnyj-kurs-otladka-programm-chast-1.html>

Все сказанное там справедливо и тут. Разве что с рядом ограничений – например в JTAG режим мы можем поставить всего три брейкпоинта.

Ошибки в работе JTAG и борьба с ними

Делятся они на два вида. Первая – студия не находит адаптер. Так и пишет, что мол не может подсоединиться. Тут надо проверить, чтобы СОМ порт был нужного номера (от 1 до 4) и правильно включены все переключатели (см. начало статьи). А также проверить правильность работы драйвера. Иногда полезно бывает взять и передернуть питание платы. Чтобы сбросить контроллер JTAG'а. Бывает что он виснет при загрузке.

Вторая ошибка – Студия видит JTAG, но говорит, что не видит Target. Тут надо проверить, контакты JTAG интерфейса, что нет нигде залипаний и неконтакта в четырехжильном проводе. А также, что активен Fuse bit JTAGEN (по умолчанию он включен, но мало ли).

Если что то не так с отладочным интерфейсом – пиши на форум, разберемся.