Книга по работе с WinAVR и AVR Studio

Роман Абраш
г. Новочеркасск
E-mail: **arv@radioliga.com**

Продолжение. Начало в №1-7/2010

ПРОБЛЕМЫ И ОГРАНИЧЕНИЯ

Наибольшее количество проблем и ограничений имеет встроенный в AVR Studio эмулятор. Все они описаны в соответствующих сопроводительных файлах, но на английском языке, поэтому здесь приведены описания наиболее важных моментов.

1. Полностью или частично не эмулируются встроенные во все типы микроконтроллеров периферийные аналоговые устройства, а так же TWI и USI.

2. Полностью отсутствует поддержка эмуляции «теневых» регистров у всех микроконтроллеров. Это проявляется, в частности, в том, что в режимах Fast-PWM и Phase-correct PWM значение регистра OCR не обновляется при достижении счетчиком TCNT верхнего значения.

3. Не поддерживается эмуляция регистра ASSR, т.е. невозможна корректная отладка программ, использующих асинхронные режимы таймера.

4. Не для всех типов микроконтроллеров поддерживаются корректно обращения к 16-битным регистрам (в частности, не эмулируется «защелкивание» значения после чтения младшего байта).

5. Некоторые биты в регистрах, которые должны сбрасываться в ноль при записи 1, могут сбрасываться и при записи 0.

6. Эмулятор выводит сообщение всякий раз, когда осуществляется попытка выполнить инструкцию, не поддерживаемую выбранным микроконтроллером, что делает отладку практически невозможной. Отключить это невозможно, кроме как исправить программу пользователя.

7. В окне Watch значения массивов обновляются не всегда. Чтобы обновить значения массива, наблюдаемого в этом окне, необходимо выполнить «сворачивание» и последующее его «разворачивание».

8. Некоторые типы переменных не отображаются корректно (например, long long).

9. Не реализовано разделение секций кода для эмуляции режимов записи в память программ.

10. Поддерживается только один режим «сна» – Idle mode.

11. WDT поддерживается не для всех моделей микроконтроллеров. Для некоторых моделей корректные интервалы WDT реализовываются только для тактовой частоты 1 МГц. В некоторых случаях при истечении интервала WDT программа не останавливается на точке останова по вектору сброса.

12. Эмулятор допускает запись в регистры PINx, причем записанное значение сохраняется там.

13. Эмулятор некорректно обеспечивает работу с портами ввода-вывода, в которых физически недоступна часть битов – все 8 битов могут использоваться в программе.

14. Регистр UDR модуля USART (UART) не может быть модифицирован никаким способом «извне» – ни вручную пользователем, ни при помощи стимуляции портов.

15. Корректная работа с парой «совмещенных» регистров USRC и UBRRH возможна только в том случае, если запись в UBRRH осуществляется только после записи в UBRC.

16. Для всех микроконтроллеров не реализована эмуляция «удвоенной» скорости SPI.

17. Не реализовано отключение периферии при помощи регистра PRR – и «отключенная» периферия продолжает эмулироваться нормально.

Все эти «нюансы» необходимо учитывать при отладке. Многие из них присущи только определенным типам микроконтроллеров – уточнить это можно, лишь обратившись к справочному файлу.

Кроме этих, имеется ряд «глюков» самой AVR Studio – например, иной раз эта среда неожиданно выгружается без каких-либо сообщений. Повторный запуск позволяет продолжить работу, как ни в чем не бывало. К сожалению, автору неизвестны ни условия возникновения этих ситуаций, ни методы борьбы с ними – все происходит необъяснимо случайно, но, к счастью, крайне редко.

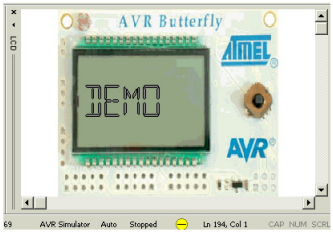
Для каждого аппаратного отладчика-эмулятора имеются свои отдельные ограничения и «нюансы» – они перечислены в соответствующей документации.

Дополнительные средства

Существует ряд бесплатных утилит, значительно облегчающих процесс написания и отладки программ для микроконтроллеров AVR. Практически все они сделаны энтузиастами, хотя имеются и фирменные.

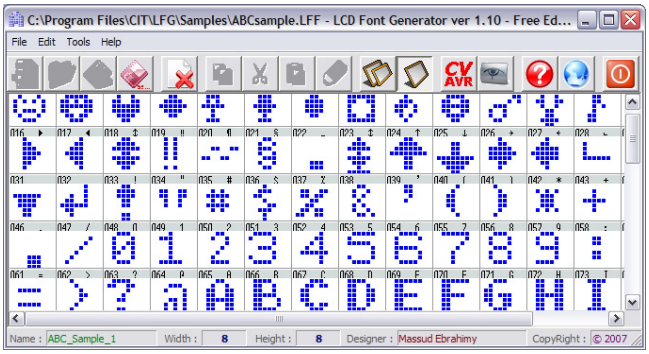
Поддержка LCD-индикаторов

Фирма Atmel бесплатно предоставляет утилиту для «визуализации» работы с LCD дисплеями микроконтроллеров со встроенными драйверами ЖКИ. Это средство позволяет разработать собственный «виртуальный» индикатор, «подключить» его к микроконтроллеру (точнее, непосредственно к его драйверу) и проводить отладку, наблюдая за тем, как работает индикатор:



Данное средство ориентировано на имитацию работы комплекта разработчика AVR Butterfly, однако может использоваться и без него.

Чаще требуется применить графический ЖКИ с встроенным контроллером, которых в настоящее время выпускается большое количество. И основная проблема, с которой сталкивается программист в этом случае – это отсутствие готовых шрифтов для вывода на такие индикаторы текстовой информации. Решение в этом случае может заключаться в использовании генератора шрифтов **LCD Font Generator**:



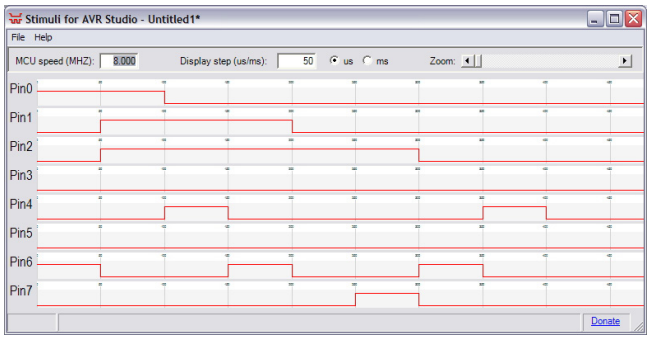
Эта утилита позволит легко создать оригинальный шрифт в виде массива констант, который затем уже можно использовать для вывода на ЖКИ. Программа ориентирована на другой компилятор Си – Code Vision CVAVR, однако получаемый код элементарно адаптируется и для WinAVR GCC.

Программа создана программистом одной из арабских стран, поэтому в некоторых случаях пытается выводить текст арабской вязью, но, тем не менее, интерфейс очень прост и удобен.

Генератор файлов симуляции внешних сигналов

Утилита создания файлов стимуляции портов **Stimuli Generator** позволяет свести сложность создания этих файлов к минимуму. Она представляет собой графический редактор импульсных последовательностей. Интерфейс программы прост и нагляден (*см. рисунок на следующей странице*):

Достаточно лишь задать в мегагерцах частоту микроконтроллера в поле **MCU speed**, установить шаг графика **Display step** и указать единицы измерения времени – микросекунды (**us**) или миллисекунды (**ms**). После этого можно «рисовать»



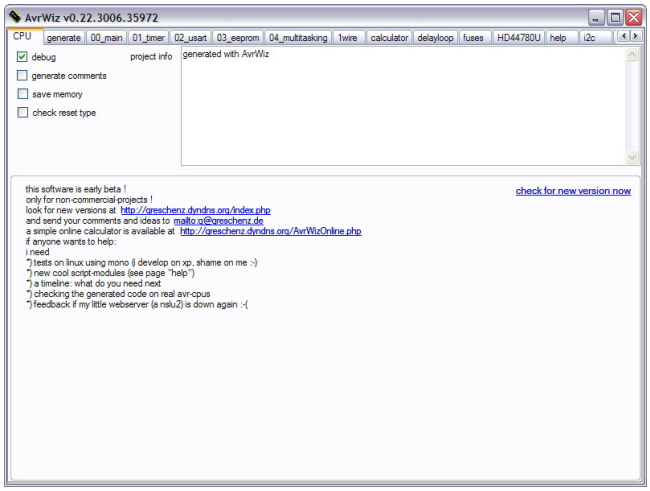
мышкой произвольные импульсы. Скроллер **Zoom** позволит увеличить или уменьшить видимую область сигнала (т.е. он меняет масштаб по оси времени).

После того, как сигналы нарисованы – надо сохранить их в файле, выполнив команду меню «**File**» **Save** или **Save As** (сохранить или сохранить с новым именем). Если необходимо – можно загрузить ранее созданный файл стимуляции командой **Load** и внести в него изменения.

Мастер создания заготовок программ

Очень мощное средство, позволяющее автоматически сгенерировать «скелет» программы в виде исходного текста на Си (и не только) – утилита **AvrWiz**.

Интерфейс программы достаточно сложный, после первого запуска окно имеет следующий вид:



Работа с утилитой заключается в том, что установкой тех или иных «галочек» активируется создание различных «заготовок» соответствующих функций и модулей программы. Помимо заготовки программы, утилита способна создать make-файл, выполнить тестовую компиляцию созданной «заготовки» и кое-что еще.

В верхней части окна программы имеются закладки, соответствующие разным возможностям. На закладке CPU задаются основные режимы работы, приводятся ссылки на сайт проекта и другая дополнительная информация.

На закладке Generate указываются модули и функции, которые программа должна создать автоматически (*см. рисунок в следующей колонке*):

Отметив опцию create AVR Studio project и нажав кнопку save, мы получаем уже подготовленный к загрузке в AVR Studio проект со всеми необходимыми файлами модулей, соответствующих отмеченным ниже возможностям:

· **00_main** – заготовка функции main()

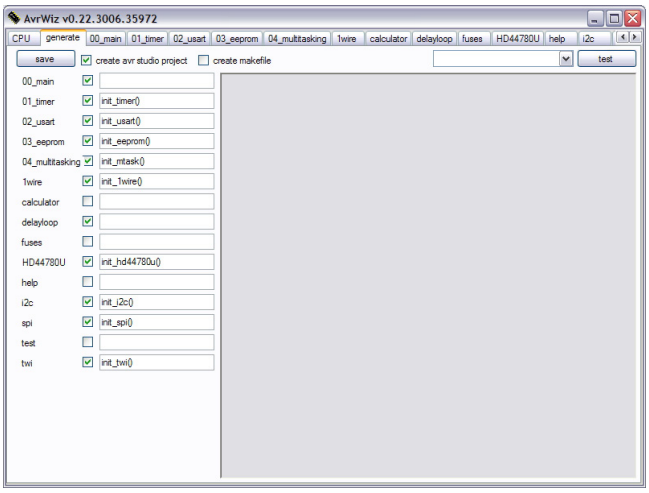
· **01_timer** – заготовки функций настройки и обработки таймеров

· **02_uart** – заготовки функций для настройки и работы с модулем UART

· **03_eeprom** – заготовки функций для работы с EEPROM

· **04_multitasking** – заготовки функций поддержки «многозадачности»

· **1wire** – заготовки функций для работы с интерфейсом 1-Wire



· **calculator** – вспомогательные вычисления

· **delay/loop** – заготовки функций программных задержек в виде циклов

· **fuses** – вспомогательные функции для правильного выбора fuse-битов

· **HD44780U** – заготовки функций поддержки ЖКИ на основе контроллера HD44780U

· **help** – справочные сведения

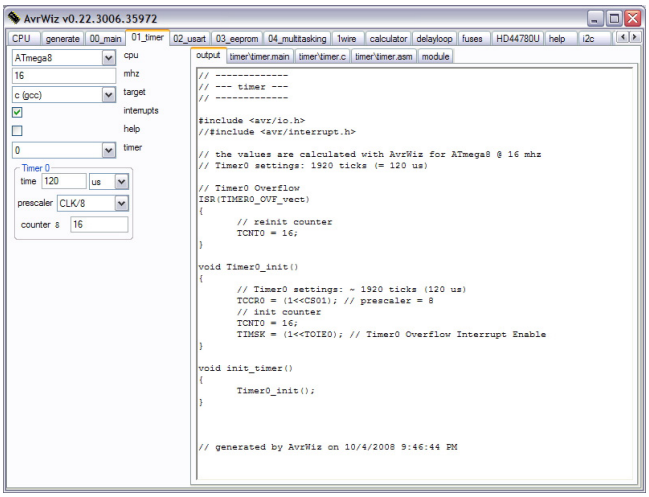
· **i2c** – заготовки функций поддержки интерфейса I²C

· **spi** – заготовки функций работы с интерфейсом SPI

· **test** – выполнение тестовой компиляции проекта-заготовки

· **twi** – заготовки функций для работы с аппаратным модулем TWI

Создание всех заготовок настраивается на соответствующих закладках, например, вот как выглядит закладка настройки таймеров:



Вверху задается тип микроконтроллера, тактовая частота в мегагерцах и указывается язык, на котором необходимо создавать исходный текст (нас интересует Си – GCC). В настоящее время поддерживается еще и ассемблер.

Ниже опциями указывается, необходима ли обработка прерываний (**interrupt**), надо ли создавать комментарии с подробным описанием кода (**help**). Затем выбирается номер таймера из числа доступных в выбранном микроконтроллере. Заключает все настройка самого режима работы таймера: можно либо ввести значение периода его переполнения (**time**) в машинных тактах (**ticks**) или интервалах времени, при этом прочие значения устанавливаются автоматически; либо указать конкретное значение предделителя (**prescaler**) и значение счетчика **TCNT** – тогда автоматически будет вычислено время его переполнения.

В большом окне слева показан код, который будет соответствовать заданным настройкам – его можно скопировать и затем вставить в свою программу, а можно воспользоваться автогенерацией проекта, как было сказано ранее.

Аналогично можно настроить и создать все прочие функции.

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ

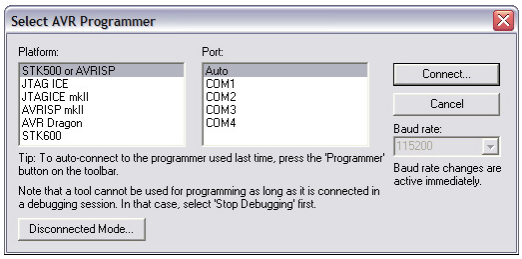
Итак, схема разработана и собрана, программа – написана и отлажена, и остается лишь вдохнуть жизнь в собранное устройство, т.е. записать в память микроконтроллера результат компиляции проекта. Этой цели служат специальные программаторы, т.е. устройства (и поддерживающие их программы), способные выполнить запись файла в микроконтроллер.

В комплексе с AVR Studio наиболее удобно пользоваться средствами, предлагаемыми фирмой Atmel или другими, совместимыми с ними, т.к. они интегрируются в среду. Однако, часто не менее удобно, а порой и значительно удобнее использовать программаторы и программы сторонних разработчиков, которых великое множество, причем многие из них совершенно бесплатны. Плюсом этого варианта следует признать возможность самостоятельной сборки схемы программатора, в то время как фирменные средства придется покупать.

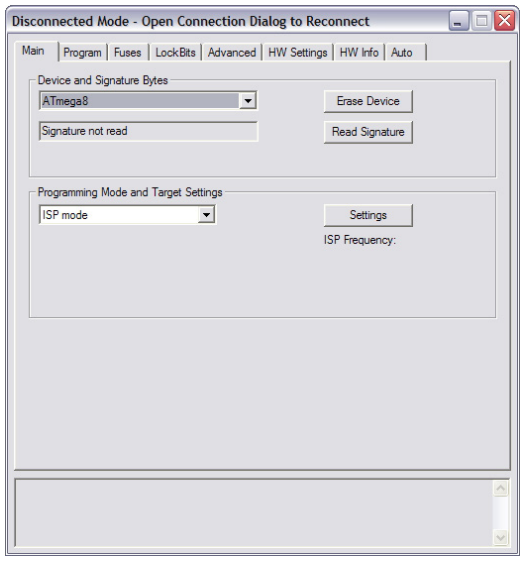
Использование встроенных средств AVR Studio

AVR Studio поддерживает следующие типы средств, способных осуществлять программирование микроконтроллеров: STK500, AVRISP, AVR Dragon и некоторые другие. Для всех этих средств используется единый интерфейс.

Подключение соответствующего программатора осуществляется командой меню «Tools» **Program AVR – Connect** или **Program AVR – Autoconnect**. Первая команда позволяет указать вручную порт, к которому подключен программатор, а вторая автоматически перебирает все подходящие порты, осуществляя поиск подходящего устройства. Если автоматический поиск неудачен – открывается окно ручного подключения (то же самое, как и для первой команды):



В левом списке **Platform** перечислены все поддерживаемые программаторы, а в правом – все порты, которые поддерживаются выбранным программатором. **Auto** – означает автопоиск порта. Выбрав платформу и порт, следует нажать кнопку Connect - произойдет подключение программатора. Чтобы поиск и подключение были успешными, следует заранее подать питание на программатор (если это предусмотрено). Если подключение успешно, то откроется окно программирования микроконтроллера (открыть это окно для ознакомления без возможности реальной работы с программатором, т.е. без подключения его, можно кнопкой **Disconnected Mode** – отключенный режим):



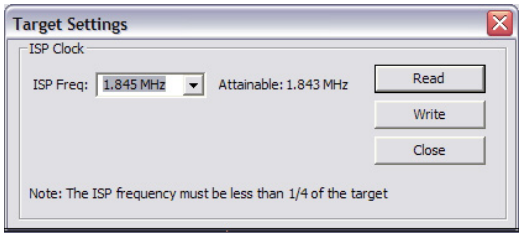
[30] Приведено окно для STK500 и совместимых с ним программаторов в «отключенном» режиме, для других типов программаторов содержимое некоторых закладок может отличаться.

Это окно по традиции содержит[30] несколько закладок-страниц, а на каждой из них ряд опций и органов управления и настройки режимов работы программатора.

Параметры программирования – Main

На закладке **Main** (основные) позволяет указать тип микроконтроллера, который планируется программировать – верхний список в группе **Device and Signature Bytes** (устройство и сигнатурные байты). Кнопка **Erase Device** выполняет полное стирание всей памяти микроконтроллера, а кнопка **Read Signature** позволяет считать сигнатуру реально подключенного к программатору микроконтроллера и сравнить ее с правильной для выбранного типа. Если сигнатуры не совпадают – все функции программирования блокируются.

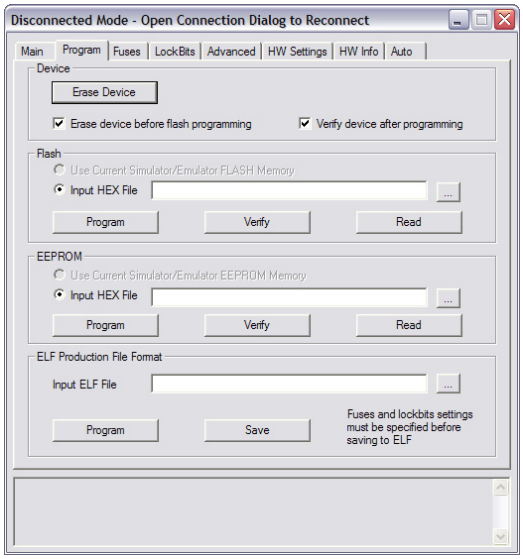
Группа **Programming Mode and Target Settings** (режим программирования и настройки) позволяет указать режим программирования: **ISP mode** (последовательное внутрисхемное программирование) или **PP/HVSP mode** (параллельное или высоковольтное последовательное программирование). Разумеется, соответствующий режим должен поддерживаться программатором. Кнопка **Settings** позволяет настроить скорость (точнее – частоту) обмена информацией с микроконтроллером при последовательном внутрисхемном программировании. Нажатие этой кнопки открывает следующее окно:



В этом окне из единственного открывающегося списка следует выбрать наиболее подходящую частоту обмена. Выбирать ее следует исходя из того, какой частотой тактируется прошиваемый микроконтроллер – частота для ISP-программирования должна быть минимум в 4 раза меньше тактовой частоты контроллера, несоблюдение этого правила может привести к невозможности программирования. Кнопка **Read** позволяет считать значение текущей частоты работы программатора, кнопка **Write** производит запись в программатор нового значения частоты.

Функции программирования – Program

Закладка Program имеет следующий вид:



Здесь предоставлены все доступные функции программирования, рассмотрим их сверху вниз по порядку.

Группа **Device** содержит кнопку полного стирания микроконтроллера **Erase Device**, а так же 2 опции, определяющие поведение программатора при выполнении остальных функций:

· **Erase device before flash programming** – стирать контроллер перед записью программы

· **Verify device after programming** – проверять запись после программирования

Назначение этих опций очевидно.

Далее имеются две очень похожих группы **Flash** и **EEPROM**, содержащих средства программирования соответственно памяти программ и данных микроконтроллера. Каждая из групп содержит одинаковые опции выбора источника данных:

· **Use Current Simulator/Emulator Memory** – использовать текущее содержимое памяти эмулятора или симулятора

· **Input HEX File** – входной HEX-файл

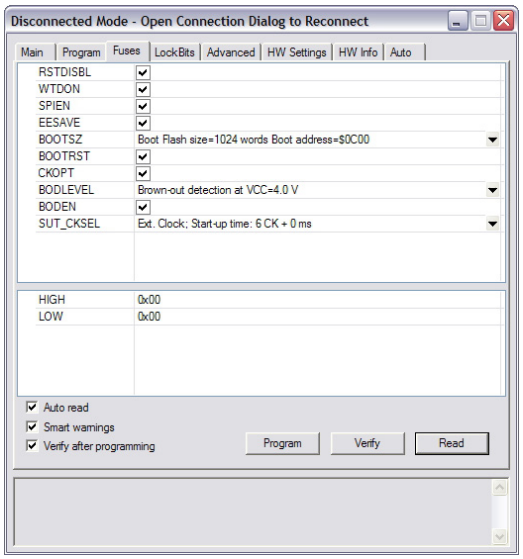
Первая опция используется в случае использования аппаратного средства отладки, которое может выполнять функции программатора, вторая – во всех остальных случаях.

Три кнопки **Program** (программирование), **Verify** (проверка) и **Read** (считывание) выполняют соответствующие функции. Считывание используется для записи в указанный файл содержимого из соответствующей области памяти.

Последняя группа содержит средства работы с файлами в формате «**elf**». В отличие от формата **HEX**, файл этого формата может содержать одновременно данные для памяти программ и для данных, а так же битов защиты и конфигурации (см. далее), поэтому программирование обеих областей осуществляется сразу.

В самой нижней области в текстовом формате выводятся сообщения о ходе выполнения соответствующих функций.

Установка fuse-битов – Fuses



На этой закладке осуществляется конфигурирование встроенных аппаратных средств микроконтроллера при помощи так называемых **fuse**-битов.

В большом верхнем окне перечислен список всех доступных для текущего режима программирования и выбранного микроконтроллера битов, а в нижнем представление этих же битов в виде, пригодном для записи в контроллер.

Далее следуют три опции:

· **Auto read** – автосчитывание значений **fuse**-битов. Если активировано, то при переходе на эту закладку состояния **fuse**-битов будут считаны из контроллера и показаны в окне.

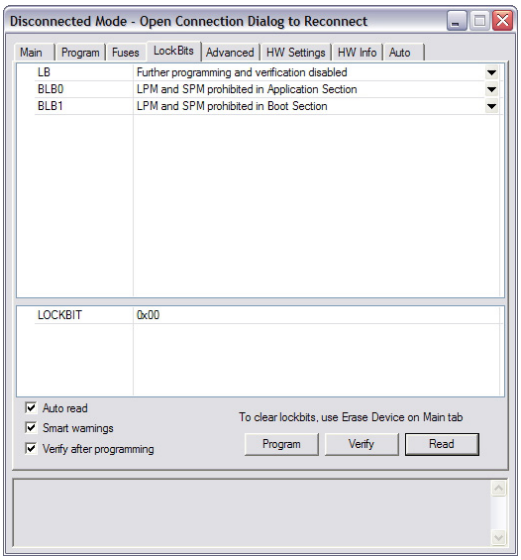
· **Smart warning** – «умные» предупреждения. Если активировано, то в случае задания опасных или несовместимых с другими значений **fuse**-битов, будут выведено соответствующее предупреждение.

· **Verify after programming** – проверять после записи.

К установке **fuse**-битов следует подходить с особой тщательностью, т.к. для режима последовательного внутрисхемного программирования существуют такие комбинации, программирование которых сделает невозможной любое последующее обращение к микроконтроллеру из программатора. В технической документации на каждый микроконтроллер fuse-биты подробно описаны, в рамках этой книги они не рассматриваются.

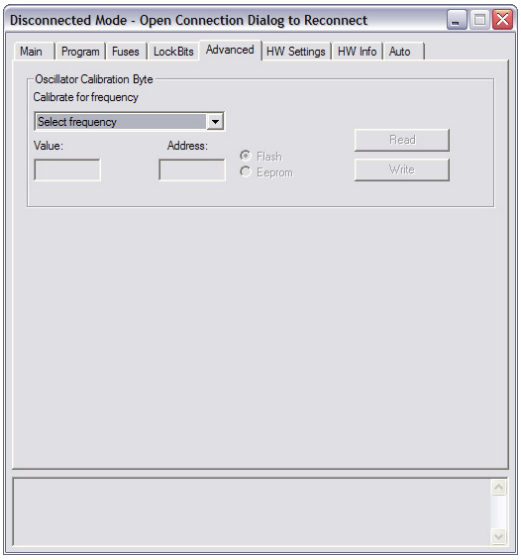
Важно помнить, что в данном случае отмеченные галочкой **fuse**-биты соответствуют *запрограммированным* (т.е. активированным).

Установка защиты – LockBits



На данной закладке задаются биты защиты памяти от несанкционированного считывания. Описание вариантов приводятся в документации к микроконтроллеру. Единственное, о чем следует помнить, это то, что установка и считывание значений битов защиты возможна всегда, а снятие – только при полном стирании микроконтроллера.

Дополнительные опции – Advanced



На данной закладке присутствуют дополнительные опции программирования. В частности, можно считать значения калибровочных байтов встроенных RC-генераторов и записать их затем в соответствующую ячейку памяти программ или данных.

Для чтения калибровочного байта следует выбрать соответствующий генератор из списка **Calibrate for frequency** и нажать кнопку **Read** – в окне **Value** будет показано считанное значение. Если указать в окне **Address** адрес ячейки памяти в памяти программ (активна опция **Flash**) или EEPROM (активна опция **Eeprom**), то после нажатия кнопки **Write** это значение будет записано в соответствующую ячейку.

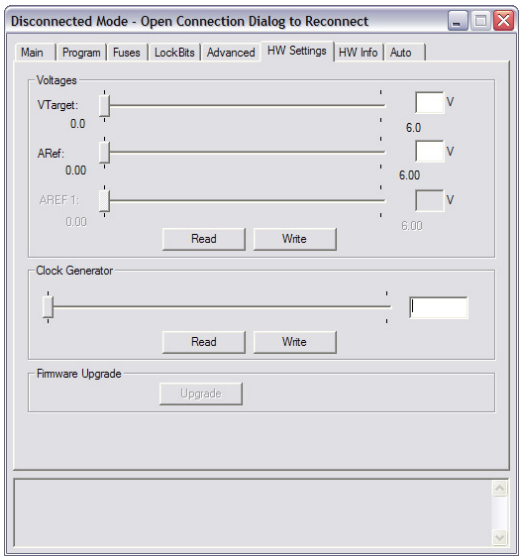
Параметры аппаратуры программатора – HW Settings (*см. рисунок на следующей странице*):

На этой закладке присутствуют органы настройки напряжений, используемых программатором для различных режимов работы, а так же других параметров. Доступность этих регуляторов зависит от модели программатора.

В группе **Voltages** имеется 3 регулятора напряжения:

· **VTarget** – напряжение питания схемы с микроконтроллером

· **ARef** – опорное напряжение для аналоговых цепей, формируемое аппаратурой (для программаторов совмещенных с отладчиком-эмулятором)



· **AREF1** – дополнительное опорное напряжение (см. документацию к аппаратному средству).

Кнопка **Read** позволяет считать текущие значения напряжений из программатора/отладчика, а кнопка **Write** осуществляет запись новых значений.

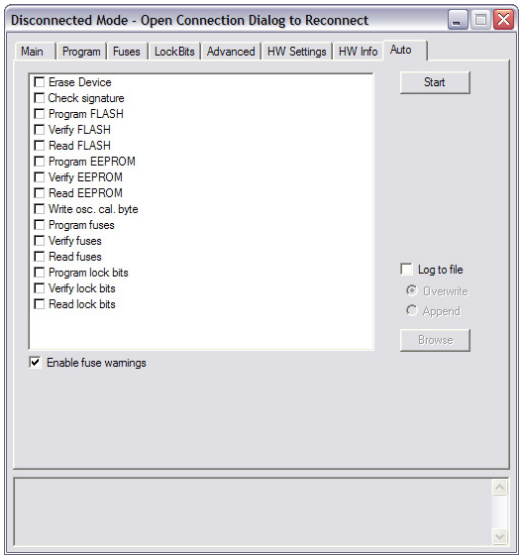
Регулятор **Clock Generator** позволяет изменить тактовую частоту контроллера программатора или отладчика.

Наконец, кнопка **Upgrade** в группе **Firmware Upgrade** позволяет обновить прошивку программатора/отладчика.

Сведения о версии – HW Info

На этой закладке (рисунок не приводится) присутствуют две строки со сведениями о версии программного и аппаратного обеспечения используемого программатора/отладчика. Эти данные используются, если возникает необходимость в проверке наличия обновлений соответствующих средств.

Автоматизация работы – Auto



На данной закладке можно задать последовательность действий, выполняемых автоматически, что может быть полезно при программировании серии микроконтроллеров. Для этого всего лишь необходимо отметить в списке те действия, которые необходимо выполнять, после чего нажать кнопку **Start** – отмеченные задачи будут выполнены последовательно сверху вниз. Если на каком-то этапе произойдет ошибка – дальнейшие действия не будут осуществлены.

Опция **Log to file** (вести протокол работы в файле) используется, если необходимо формировать текстовый файл, содержащий сведения о ходе каждой операции. Если каждый раз файл следует перезаписывать, нужно активировать опцию **Overwrite**, а если следует дописывать к уже имеющемуся файлу – **Append**. Выбрать месторасположение и имя файла можно, нажав кнопку **Browse**.

Использование средств сторонних разработчиков

Помимо встроенных в AVR Studio средств программирования, ориентированных на применение достаточно дорогого фирменного аппаратного обеспечения, существует большое количество бесплатных любительских средств, зачастую не менее функциональных. К сожалению, они не интегрируются в среду AVR Studio, а выполнены в виде отдельных программ. Аппаратное обеспечение для них обычно крайне простое, в некоторых случаях состоит всего из 4-6 деталей, но оказывается достаточным для многих случаев.

PonyProg

Наиболее известна программа **PonyProg** итальянского автора Клаудио Ланконелли. Программа использует несложные адаптеры, подключаемые к COM или LPT портам компьютера, и умеет работать практически со всеми микроконтроллерами AVR, а так же с большим количеством других микроконтроллеров и программируемых микросхем. Имеется версия программы с русифицированным интерфейсом, что является несомненным ее достоинством.

PonyProg осуществляет только внутрисхемное последовательное программирование микроконтроллеров, режимы высоковольтного и параллельного программирования не поддерживаются.

В настоящее время существует довольно большое количество недорогих «клонов» фирменных программаторов типа STK500, прибрести которые можно во многих интернет-магазинах. Однако, бóльший интерес для любителя может представлять клон, который можно изготовить самостоятельно – это, например, AvrUsb500, разработанный автором Petka (это интернет-ник). Об этом программаторе можно более подробно узнать в интернете по адресу:

http://electronix.ru/forum/index.php?showtopic=68372.

Этот клон фирменного комплекта разработчика STK500 является полностью совместимым с AVR Studio, которая опознает его как родной STK500. Следует только не забывать, что режимы «высоковольтного» и параллельного программирования этот клон (да и большинство других) не поддерживает.

По ключевым словам «клон STK500» можно найти в интернете и другие варианты подобных программаторов, доступных для самостоятельной сборки.

Продолжение в №9/2010