APU-2

USB-программатор для микроконтроллеров AVR фирмы Atmel

podkassetnik@yandex.ru

Историческая справка, общие сведения

Не секрет, что микроконтроллеры AVR фирмы Atmel до сих пор пользуются достаточной популярностью у радиолюбителей. Да и у профессионалов тоже, поскольку для перехода на новые, более прогрессивные типы чипов (те же STM32F от ST Microelectronics), обычно просто тупо не хватает времени. Вроде и желание есть, и собственные финансы тратить нет необходимости (контора всё оплатит, ага) – а вот заняться детальным изучением и внедрением некогда. Особенно, когда ведешь четыре-пять проектов одновременно, а сдавать каждый надо «ещё вчера». Поэтому, как говорится – и хочется, и колется.

С микроконтроллерами AVR мне пришлось столкнуться по профессиональной нужде. В «далеком» 2000-м году, когда я, еще учась в институте, только устроился работать по своей специальности (Радиотехника), все дывайсы нами изготавливались из «рассыпухи» – микросхем тупой логики а-ля ЛА3, ТМ2, ИЕ7 и т.д., а также мега-памяти типа К556РУ5, К556РТ4 и прочих. При этом размеры печатных плат средненького устройства получались – мое почтение. Правда, и корпуса для радиоаппаратуры в то время были подстать платам: просторные до ужаса (от ширины душевной) и неубиваемые (потому что из металла).

Однако время шло, и функционал разрабатываемых дывайсов постоянно усложнялся. К тому же, хитрые заказчики постоянно норовили увязать устройства с компом. Наиболее доступен и удобен в использовании в то время был LPT-порт (особенно из-под Windows 98). Ну и как-то так получилось, что на фоне лавинообразного усложнения функционала устройств он стал выглядеть, мягко говоря, бледновато. Одно время пытались делать платы на ISA-шную шину, но как-то это дело быстро заглохло – мало места для полета конструкторской фантазии, да еще платы получались минима двухсторонние с кучей дополнительных проводочков-перемычек. Нужен был COM-порт (про USB в то время еще мало кто задумывался). Однако, чтобы связать дывайс с COM-портом на «рассыпухе» надо было неслабо поплясать с бубном (знающие люди не дадут соврать).

Поскольку я кроме всего прочего отвечал за разработку и изготовление печатных плат (ПП), мне такое положение вещей стало вполне естественным образом надоедать. А тут еще знакомые-искусители периодически подтаскивали на разработку/изготовление прототипов ПП свои проекты на основе микроконтроллеров AT90S. По словам авторов, функционал данных дывайсов обычно на порядок превосходил наши, но самое главное – они могли запросто общаться с компом по COM-порту, а ценой за это было использование ширпотребовских микросхем MAX232 за смешные деньги.

Глядя на изготовленные для знакомых лаконичные прототипы печатных плат, я их все время сравнивал с лежащими неподалеку нашими «шедеврами» размером с тетрадный лист (или кирпич – в зависимости от проекта). И в какой-то момент решил – необходимо самому осваивать микроконтроллеры. К тому же, после красочного изложения открывающихся перспектив начальство такую инициативу молодого специалиста пообещало даже проспонсировать.

Итак, я приступил к изучению микроконтроллеров AVR. Было это где-то в 2005-2006 году. По тем временам у меня был нехеровый козырь – наличие безлимитной выделенки на работе. В то время как большинство знакомых через модем качало по пять песен за ночь, я скачивал по два-три альбома за пару часов (это потому, что у меня был доступ к нескольким компам). И вот вдруг внезапно оказалось, что Интернет на работе нужен не только для скачивания песен/фильмов. Оказалось, что из него можно почерпнуть много полезного непосредственно по работе. Я выяснил, что армия поклонников AVR неумолимо растет ужасающими темпами. Также узнал, что помимо покупки самого камня (т.е. микроконтроллера) неплохо бы было озаботиться

приобретением программатора для него. Ибо без программатора камень так и останется камнем, а мне нужен был работающий дывайс.

В г. Горький в то время можно было приобрести за сравнительно небольшую сумму программатор STK300. Но! Его пришлось бы ждать пару недель, а мне хотелось «всего и сразу». Поэтому я для начала остановился на программаторе «5 проводков» для LPT-порта, пользующемся в ту пору бешеным спросом у радиолюбителей. Основное его преимущество – полнейшее отсутствие в нем радиокомпонентов (провода за таковые не считаю). Конечно, была весьма ненулевая вероятность сжечь порт компьютера, но во-первых, я привык все делать аккуратно, а во-вторых, компов в моем распоряжении было несколько, так что один бы точно дожил до первой удачной прошивки. Результатом эксперимента я был более чем доволен – всеми любимая «лошадь» считала и прошила фузы на ATMega8515 с первого раза. В последствии я себе сделал также программатор LPT с буферными элементами, а также собрал несколько программаторов для COM-порта (расходились по знакомым как горячие пирожки).

Таким образом, в те времена при наличии Интернета от начала желания что-либо залить в микроконтроллер до начала непосредственно прошивки камня у новичков в среднем уходило пара-тройка часов. При этом, насколько я могу судить, подавляющее большинство первых экспериментов проходило удачно. Ну а если же все-таки были какие-то проблемы, найти их причину обычно не составляло труда, поскольку все детали были просты, как барабан, и предсказуемы, как стадо. Но в последнее время все несколько изменилось. И связаны эти изменения как с вытеснением COM, и особенно LPT портов портом USB, так и со всё увеличивающимся объемом FLASH-памяти микроконтроллеров.

Сейчас самая убогая материнка содержит 8-10 портов USB, да не просто USB, а минима USB2.0. Зато COM (и особенно LPT) порты присутствуют на борту матери далеко не всегда. И человеку, решившемуся на освоение микроконтроллеров, часто приходится выбирать: или менять компьютерное железо в погоне за простотой программатора, или попытаться собрать программатор для USB порта. Обсудим сначала первый вариант (закупка новой материнки с COM и/или LPT портом).

Основная фича «древних» программаторов (это которые для LPT и COM портов) заключается в том, что (как было сказано выше) все используемые в них детали являются, так сказать, самодостаточными. Т.е., если в программаторе и используются какие-то микросхемы, то это или обычная логика, или буферные элементы. Эти детали (наряду с резисторами, конденсаторами, транзисторами и разъемами) нет нужды прошивать перед использованием, они работают сразу после монтажа платы в соответствии со своим предназначением. Т.е., собрал программатор – и сразу можно прошивать камень (ну, максимум – еще подать напряжение питания на программатор). И это делало микроконтроллеры, в частности AVR-овские, весьма привлекательными (особенно для начинающих). Раньше (когда COM и LPT порты присутствовали практически на каждом компьютере) обычно посты о первом знакомстве с AVR выглядели примерно так: «Пацаны, купил сегодня на рынке микруху avr, воткнул ее в лпт через проводки, все залилось, все работает, КРУТОООООООО!!!!!!!!!!!». Ну, или на крайняк: «Собрал на коленке программатор из того говна, что валялось на столе, купил камень, все залилось, все работает, РУУУУЛЛЛЛЕЕЕЕЕЕЕЕЗЗЗЗЗ!!!!!!!!!!!». А причина простоты и надежности «древних» программаторов так же проста, как и они сами. При использовании LPT или COM порта прошивающий компьютер просто тупо генерит нужную последовательность электрических импульсов для линий SCK, MOSI, MISO и RESET микроконтроллера на определенных контактах соответствующего порта. А задача программатора состоит лишь в том, чтобы подать эти сигналы на нужные выводы камня с соответствующим уровнем сигнала (так, программаторы для COM порта переделывали +/-12В в 5/0В). Именно поэтому «древние» программаторы не используют никаких «умных» радиоэлементов – они там попросту не нужны. «Умный радиоэлемент» – это

компьютер. Однако, в такой огромной медовой бочке простоты есть также место нехилому половнику дегтя.

Дело в том, что «древние» программаторы являются относительно медленными. Скорость прошивки зависит от компьютерного железа, но все равно (опять же – относительно) невелика. По моим личным ощущениям, для микроконтроллеров с объемом FLASH-памяти до 8-16кБ быстроты LPT/COM-программаторов вполне хватает. А вот дальше (при объеме флэша 32кБ и более) тормознутость программатора (особенно для COM порта) начинает раздражать. И чем более сложный проект, чем больше циклов перепрошивки – тем больше эти тормоза бесят. Об этом следует знать людям, выбравшим первый вариант знакомства с микроконтроллерами AVR. Ибо если на начальном этапе и кажется, что 8кБ – это за глаза и уши, то нет никакой гарантии, что уже через полгода не будет мало 64кБ.

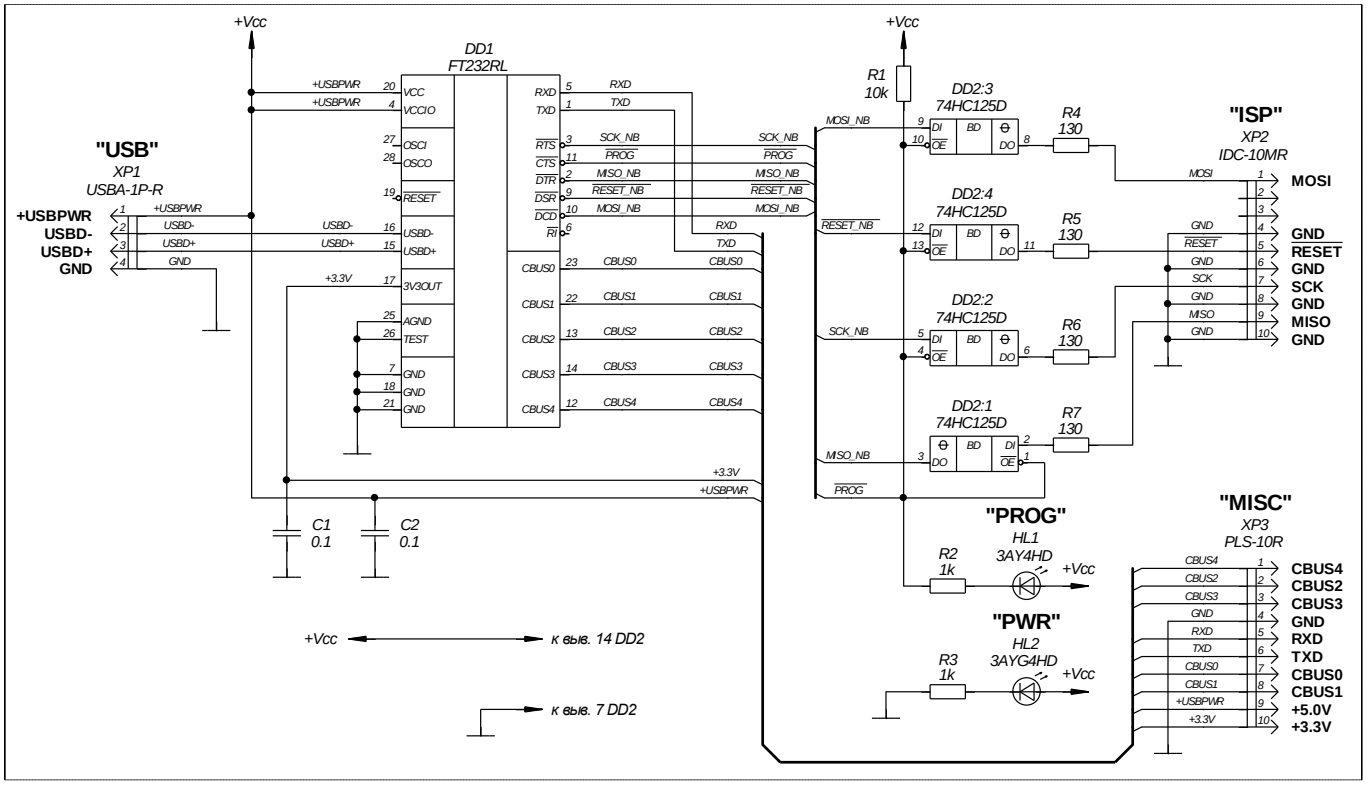
А теперь обсудим второй вариант знакомства с микроконтроллерами AVR – использование USB-программатора для прошивки камня. Правильный USB-программатор – вещь, фактически, универсальная. Его можно воткнуть в любой современный комп (а особенно – ноутбук) и без проблем перешить нужный микроконтроллер с любым объемом FLASH-памяти на довольно высокой скорости. Но ключевое слово здесь – правильный. Правильный – это который нормально работает без настройки и танцев с бубном над ним сразу же после установки и монтажа деталей. Правильный – это такой, который не глючит при переходе от одного компа к другому или смене ОС. Правильный – это такой, дрова на который есть для любой современной широко используемой версии ОС, и эти дрова неглючные. Каждый определит еще с десяток критериев правильности для себя лично, но вышеперечисленные – основные, без соблюдения которых нормально работать с микроконтроллером невозможно будет в принципе.

В настоящее время в Интернете полно различных схем USB-программаторов для AVR. Условно их можно разделить на две большие группы. Первая группа включает в себя программаторы, построенные на основе микроконтроллеров (в частности, AVR). Я собирал несколько штук программаторов от Prottoss'a (AVR910), себе и своим знакомым, а также несколько штук USBasp. Двое из знакомых, одаренных сиими дывайсами, в восторге. Удачно шьют камни в течение уже нескольких лет. У остальных (в частности – у меня лично) собранные программаторы особой радости не вызвали. Я не говорю, что они плохие, просто вот так складывались обстоятельства: на одном компе работает, на другом нет. Или, проработав пару часов, оказывались невидимыми для софта, через который шьется камень. И много еще чего. Сразу оговорю – я не разбирался с прошивкой контроллеров, на которых данные программаторы собраны. Правда, перепробовал кучу программ-прошивальщиков, через которые данные программаторы, вроде как без проблем должны шить камни. Однако, результат в виде частых глюков меня не особо удовлетворил. Исключение составила только программа AVRDUDE в комплексе с графической оболочкой SinaProg, но о ней я узнал слишком поздно ☺ (см. далее). Кстати, заметил такую тенденцию: чем древнее комповое железо, тем лучше работают данные программаторы. Ну и самый неприятный момент для тех, кто выбрал второй вариант знакомства с микроконтроллерами AVR – чтобы программатор заработал, нужно чем-то прошить входящий в его состав камень. Т.е. получается так: чтобы пользоваться программатором нужно сделать/найти программатор, чтобы прошить мозги этого программатора. Вот такой вот замкнутый круг. Вывод: себе бы сейчас я делать такой не стал. Но вовсе не потому, что программатор – говно. При удачном стечении обстоятельств вполне хорошая весчь (кстати, один раз реально спас меня – нужно срочно было перешить один автомобильный дывайс именно в машине, без съема устройства с борта авто, а кроме данного программатора и ноутбука под рукой ничего не было. Так USBasp справился с заданием на ура). Делать я его не стал бы потому, что появилось другое доступное решение, более прогрессивное на мой взгляд. И вторая группа USB-программаторов включает в себя именно это решение.

Я говорю о специализированной микросхеме FT232Rx. В свое время данная микросхема стала своего рода революцией (в узких кругах☺). Мало того, что она без особых заморочек для разработчика преобразует USB в UART (и, наверное, 95% разработчиков используют ее именно в этих целях). Она еще умеет эмулировать полноценный COM-порт, причем состояние «второстепенных» линий (таких, как RTS, CTS, DTR и т.д.) можно задать/считать не из виртуального COM-порта, а напрямую через драйвер FTDI (разработчика FT232Rx). Таким образом, появилось новое «тупое» (т.е., без необходимости первичной прошивки мозгов программатора) решение для прошивки микроконтроллеров, причем, довольно шустрое.

Железо

Схемная реализация вышеупомянутого решения приведена на *Рис. 1* (файл APU-2_SCH.pdf в архиве **APU-2_Hardware.rar**). Ничего нового здесь не придумано, никаких Америк не открыто. Описание работы схемы неоднократно приводились в Интернете (например, здесь и здесь. Или тут). В принципе, здесь и описывать-то особо нечего. Данная схема просто направляет сигналы MOSI, MISO, SCK и RESET, которые формируются на выводах DCD, DTR, RTS и DSR микросхемы DD1 (FT232RL) соответственно, на нужные выводы прошиваемого микроконтроллера (т.е., фактически является аналогом «древних» программаторов). Причем, делает это только в момент программирования камня, в остальные моменты времени программатор отключен от прошиваемой платы за счет 4-х буферных элементов микросхемы DD2 (74HC125D). Состояние линий MOSI, MISO, SCK и RESET устанавливается/считывается прошивающим софтом на компьютере. Передача данных между компом и микросхемой FT232RL идет по шине USB (от которой еще и получает питание программатор).



1. *Максимальные габаритные размеры печатной платы - 60x25мм*
2. *Токопроводящий рисунок печатной платы проектировать с учетом ГОСТ 23751-86 (класс точности - не выше 3)*
3. *Максимальное количество токопроводящих слоев печатной платы - 1*
4. *Все конденсаторы - керамические типоразмера 0805 либо 1206 с типом диэлектрика X7R или Y5V*
5. *Все резисторы типоразмера 1206 с допуском на отклонение значения сопротивления от номинального не более 5%*
6. *Разъем XP1 ("USB") - Тип A/Вилка/Угловой/B отверстия (DS1097 "Connfly")*
7. *Допускается в качестве элемента XP2 устанавливать разъем штыревого типа PLD-10R*

Рис. 1

Светодиод HL2 («PWR») сигнализирует о подаче на программатор напряжения питания с шины USB. Светодиод HL1 («PROG») индицирует процесс прошивки микроконтроллера (горит только во время прошивки). Вот, в принципе, и все описание собственно схемы электрической принципиальной. Единственное что хотелось бы отметить: во-первых, для подключения программатора к прошиваемой плате используется разъем IDC-10MR (XP2 «ISP»), распиновка которого совпадает с широко распространенной распиновкой разъема программатора STK200/STK300:

XP2 "ISP"

Разъем для подключения устройства к программируемому микроконтроллеру

#	ЦЕПЬ	ОПИСАНИЕ
1	MOSI	Линия MOSI (Master Out Slave In)
2		
3		
4	GND	Общий провод
5	RESET	Линия сброса программируемого микроконтроллера
6	GND	Общий провод
7	SCK	Линия SCK (тактовый сигнал)
8	GND	Общий провод
9	MISO	Линия MISO (Master In Slave Out)
10	GND	Общий провод

И второе (на мой взгляд, более важное) – в данной схеме на разъем XP3 «MISC» выведены все неиспользуемые выводы микросхемы FT232RL:

- **RXD** и **TXD** как самые «вкусные» для большинства;
- **CBUS0**-**CBUS4**, которые часто и в рассмотрение-то не принимают, а зря;
- выход встроенного стабилизатора на **+3,3В/50мА**

и напряжение +5,0В с шины USB:

XP3 "MISC"

Разъем для использования дополнительных функций программатора (реализуются на возможностях микросхемы DD1 FT232RL)

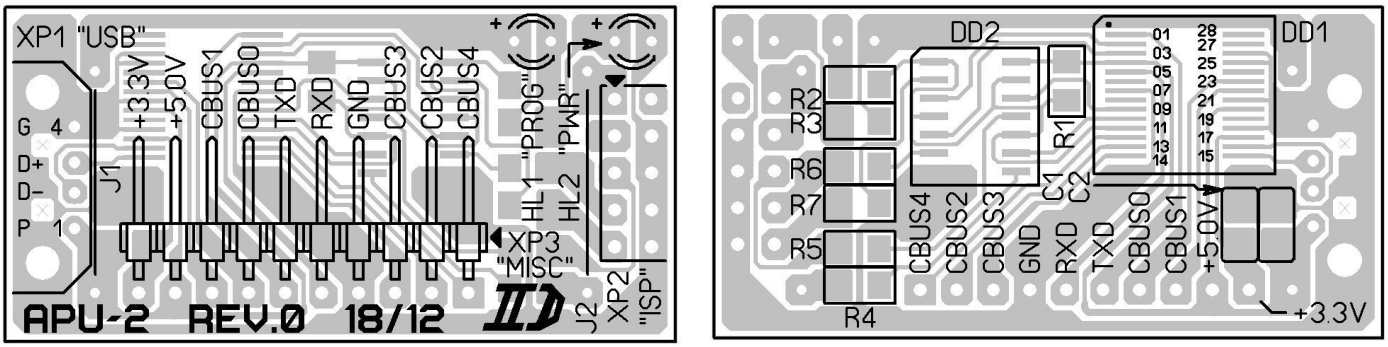
#	ЦЕПЬ	ОПИСАНИЕ
1	CBUS4	Линия CBUS4 порта ввода/вывода CBUS м/с FT232RL
2	CBUS2	Линия CBUS2 порта ввода/вывода CBUS м/с FT232RL
3	CBUS3	Линия CBUS3 порта ввода/вывода CBUS м/с FT232RL
4	GND	Общий провод
5	RXD	Вход приемника UART м/с FT232RL
6	TXD	Выход передатчика UART м/с FT232RL
7	CBUS0	Линия CBUS0 порта ввода/вывода CBUS м/с FT232RL
8	CBUS1	Линия CBUS1 порта ввода/вывода CBUS м/с FT232RL
9	+USBPWR	Напряжение +5,0В (с шины USB)
10	+3.3V	Напряжение +3,3В (с м/с FT232RL)

Сделано это со следующей целью. На мой взгляд, микросхема FT232RL имеет довольно нехилый потенциал для разработчика (например, линии шины CBUS можно использовать как обычные линии ввода-вывода микроконтроллера), поэтому неплохо бы иметь доступ ко всем ее выводам. Ну и доступ к напряжениям +5,0В и +3,3В тоже лишним никогда не будет.

Вид печатной платы программатора со стороны деталей и со стороны фольги показан на *Рис. 2а* и *Рис. 2б* соответственно. Печатная плата выполнена на одностороннем фольгированном стеклотекстолите с толщиной подложки 1,5мм и толщиной токопроводящего слоя 35мкм. Исходные файлы для изготовления ПП в формате Sprint Layout 5.0 доступны в архиве **APU-**

2_Hardware.rar (файл APU-2_ЛУТ.LAY – для «утюжников», файл APU-2_ФР.LAY – для «шаблонщиков»). Просмотрщик файлов .LAY можно скачать тут (в самом низу страницы).

Ориентировочные значения габаритных размеров печатной платы и готового программатора приведены на *Рис. 3*. Внешний вид собранного устройства показан на *Рис. 4*.



а) *б)*

Рис. 2

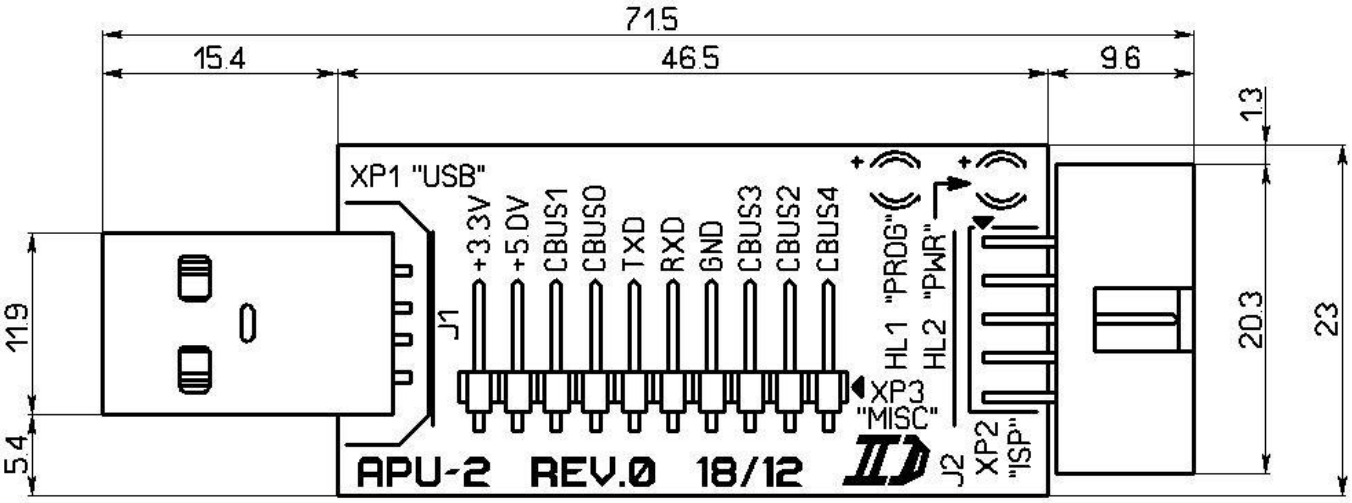
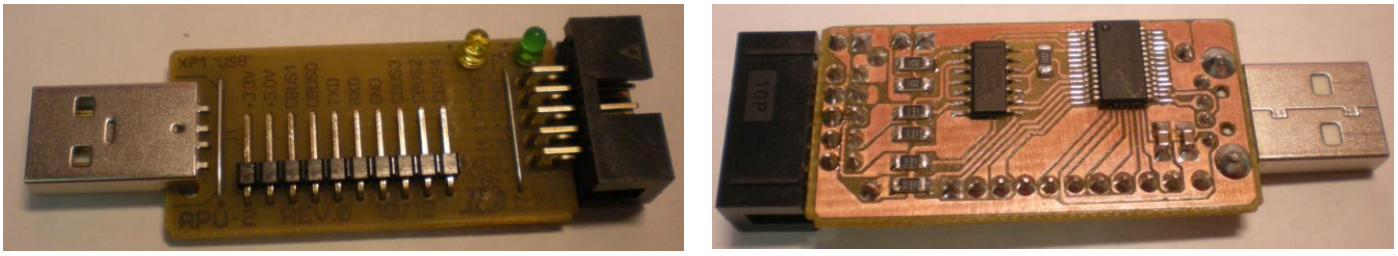


Рис. 3



а) *б)*

Рис. 4

Собранный программатор не требует никакой настройки. После установки и монтажа деталей можно сразу вставлять его в порт USB и приступать к установке драйверов и прошивающего ПО.

Софт

Для работы программатора потребуется драйвер FTDI для микросхемы FT232RL. На сайте производителя содержатся (на момент написания заметки) драйвера для следующих ОС:

Windows Server 2008 R2;
Windows 7;
Windows 7 x64;
Windows Server 2008;
Windows Server 2008 x64;
Windows Vista;
Windows Vista x64;
Windows Server 2003;
Windows Server 2003 x64;
Windows XP;
Windows XP x64;
Windows ME;
Windows 98;
Linux;
Mac OS X;
Mac OS 9;
Mac OS 8;
Windows CE.NET (Version 4.2 and greater);
Android.

Заходим на сайт FTDI, слева выбираем «DRIVERS». Далее слева откроется еще два пункта: «VCP Drivers» и «D2XX Drivers». Дрова VCP создают в компе виртуальный COM-порт, через который можно общаться с микросхемой FT232Rx как через обычный COM. Дрова D2XX позволяют достучаться до FT232Rx напрямую через DLL. Как говорится в инструкции по установке, в общем случае эти два драйвера взаимоисключающие и не могут быть установлены одновременно. Однако, для Windows 2000, XP, Vista, 7 производитель создал объединенные дрова (Комбинед Дров-Модель – Combined Driver Model (CDM)), включающие в себя и VCP и D2XX. Т.е. для перечисленных версий ОС при установке CDM на компе появляется виртуальный COM-порт, однако одновременно есть возможность прямого доступа к микросхеме через библиотеки. Можно убедиться – для 2000, XP, Висты и Семерки в случае выбора «VCP Drivers» и «D2XX Drivers» для каждой версии ОС будет скачан одинаковый архив с дровами. Надо сказать, что лично мне приходилось работать с данным программатором только на XP и Висте. Для данных версий ОС дрова можно скачивать хоть из пункта «VCP Drivers», хоть из пункта «D2XX Drivers». Про прочие поддерживаемые ОС, к сожалению, сказать ничего не могу, пользователям данных систем предлагаю разобраться в вопросе самостоятельно. В дальнейшем же буду рассматривать установку драйверов на примере Windows XP Professional SP2.

Итак, нажимаем на «VCP Drivers» или на «D2XX Drivers». Далее в таблице выбираем дрова для необходимой ОС. После скачивания распаковываем архив. Для определенности будем считать, что архив с дровами распакован в папку *D:\FTDI_DRV*.

Также потребуется программа-прошивальщик. В качестве оной рекомендую использовать ПО SinaProg (утащено из каментов отсюда, папка **SinaProg** доступна в архиве **APU-2_Software.rar**). Архив **APU-2_Software.rar** лучше распаковать прямо на диск C, чтобы путь к папке **SinaProg** был таким: *C:\SinaProg*.

Перед установкой драйверов рекомендую зайти в диспетчер устройств и посмотреть список установленного оборудования в разделах «Контроллеры универсальной последовательной шины USB» и «Порты (COM и LPT)» (*Рис. 5*):

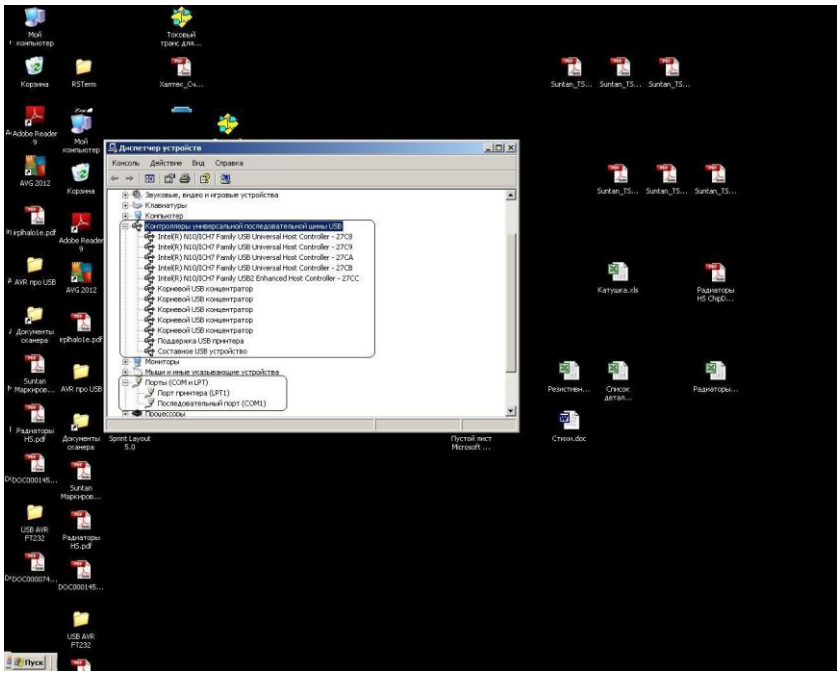


Рис. 5

Дальше необходимо вставить программатор в USB порт компьютера. Если всё смонтировано верно и на плате нет соплей, должен загореться светодиод HL2 («PWR»), а на экране появится надпись про новое оборудование и автоматически запустится соответствующий мастер (*Рис. 6*):

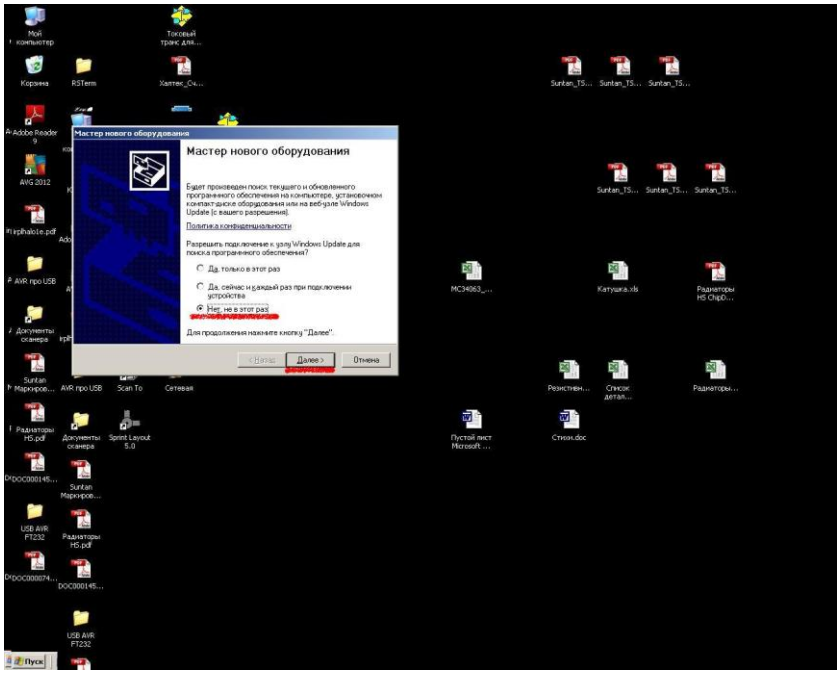


Рис. 6

Выбираем «Нет, не в этот раз» и нажимаем «Далее». На вопрос, откуда будут устанавливаться дрова, отвечаем «Установка из указанного места» и жмем «Далее» (*Рис. 7*):

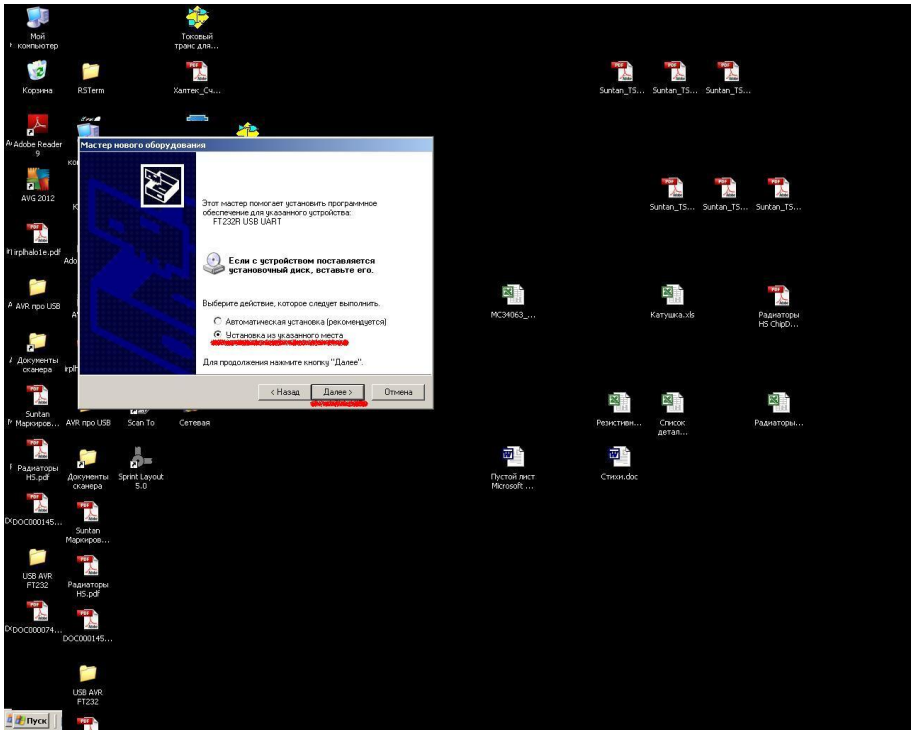


Рис. 7

Система запросит место нахождения дровяных файлов. Отмечаем «Включить следующее место поиска» и нажимаем «Обзор» (*Рис. 8*):

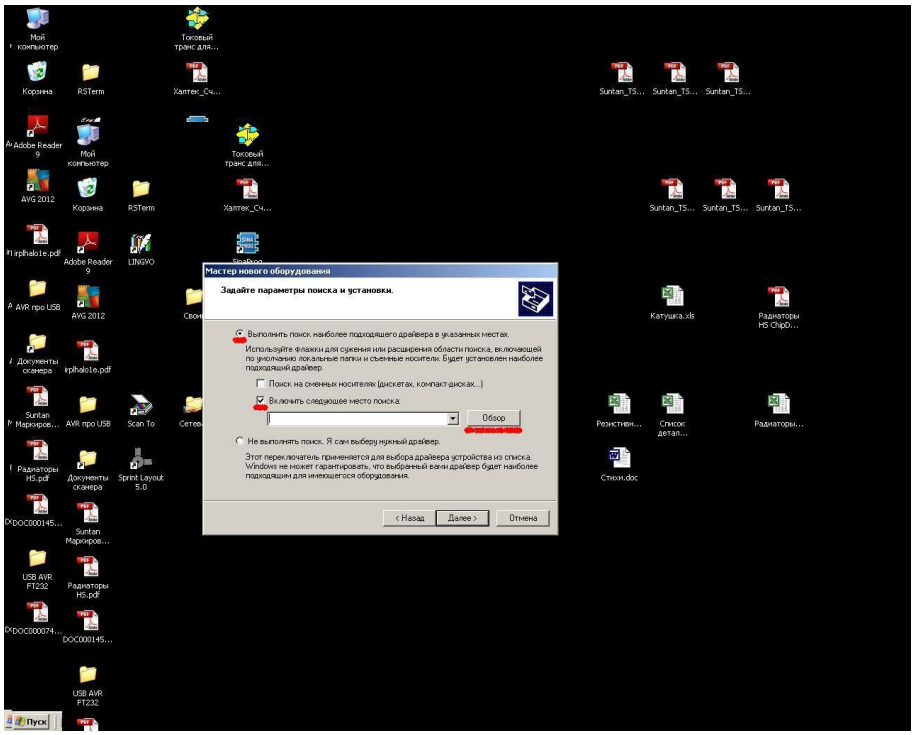


Рис. 8

В открывшемся проводнике необходимо указать путь к папке, в которую были распакованы драйвера для FT232RL (напомню, что выше для определенности мы условились, что архив с дровами распакован в папку *D:\FTDI_DRV*), после чего нажать «ОК» (*Рис. 9*):

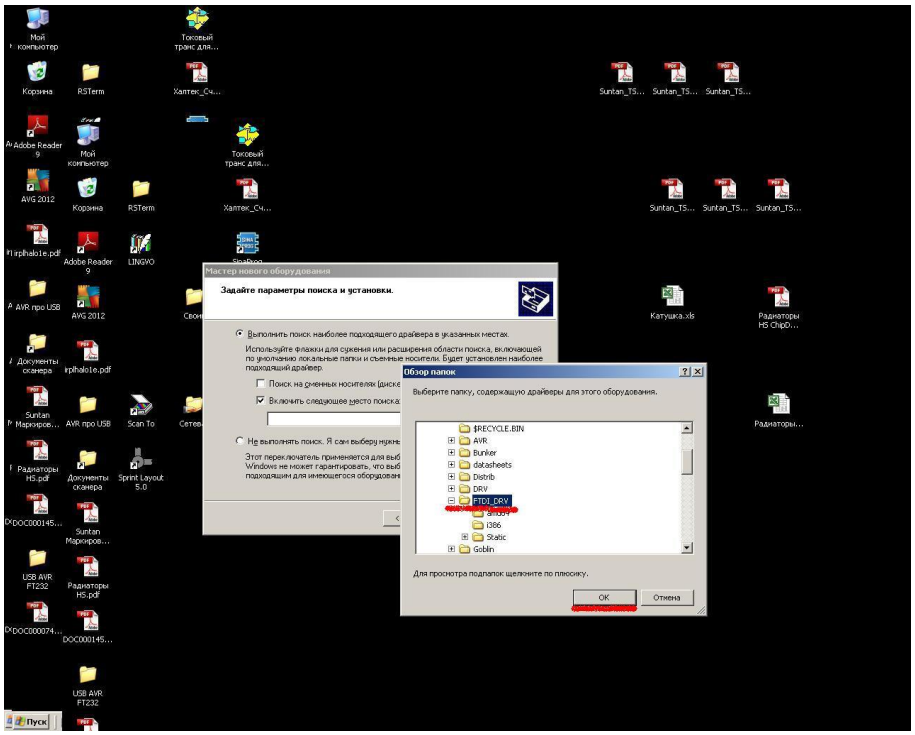


Рис. 9

Затем необходимо нажать «Далее». Если путь к папке указан верно, система приступит к установке драйверов для последовательного преобразователя USB (*Рис. 10*):

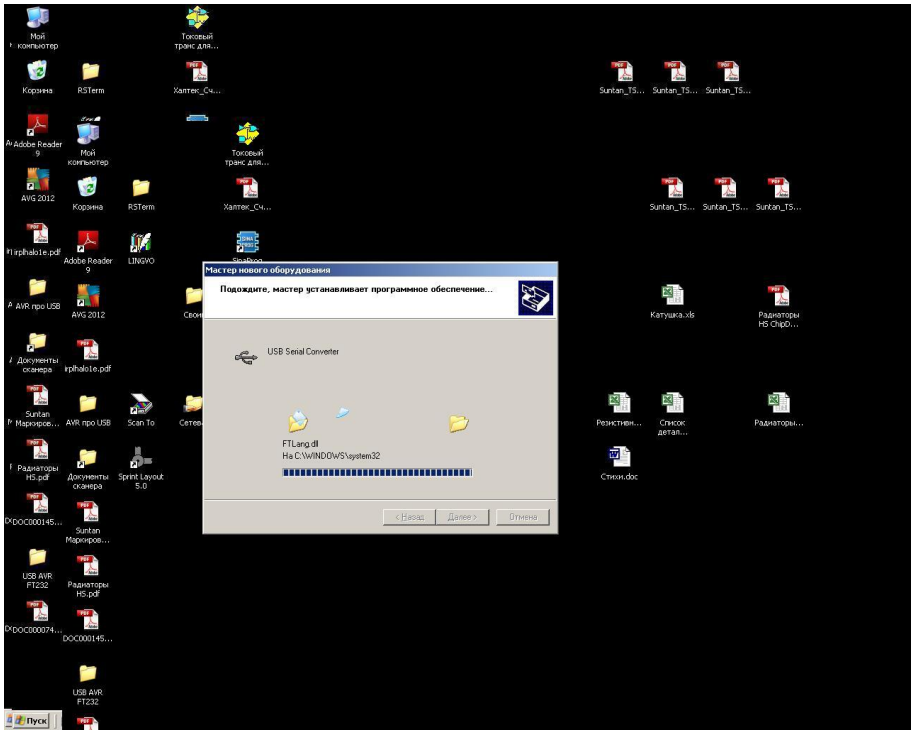


Рис. 10

И для завершения работы мастера установки нового оборудования необходимо нажать кнопку «Готово» (*Рис. 11*):

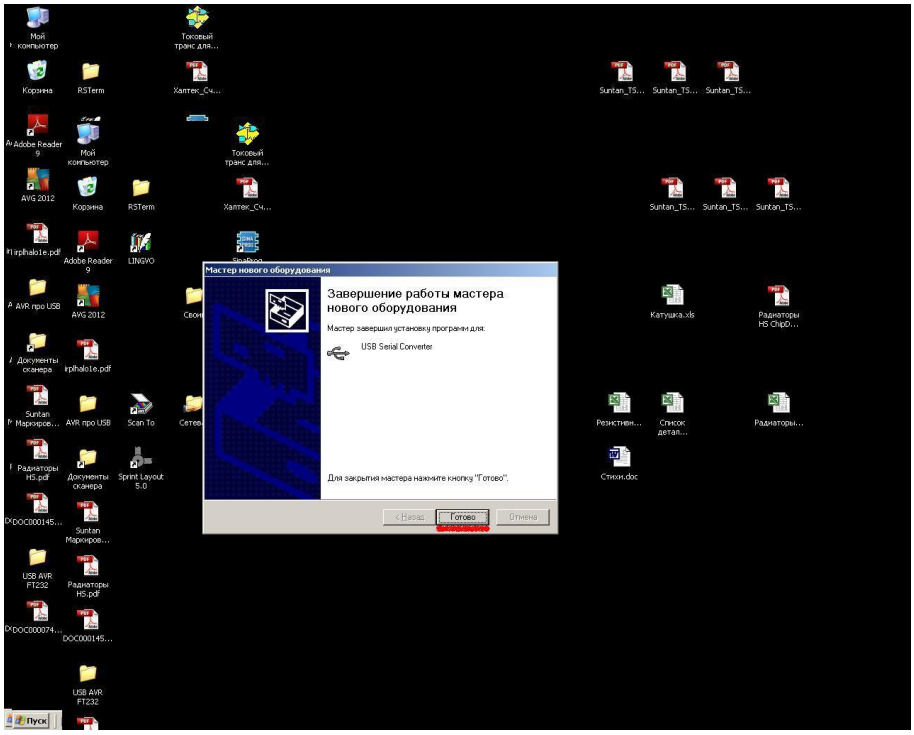


Рис. 11

Далее система сразу обнаружит еще одно новое устройство: последовательный порт USB. Снова запустится мастер установки нового оборудования (*Рис. 12*):

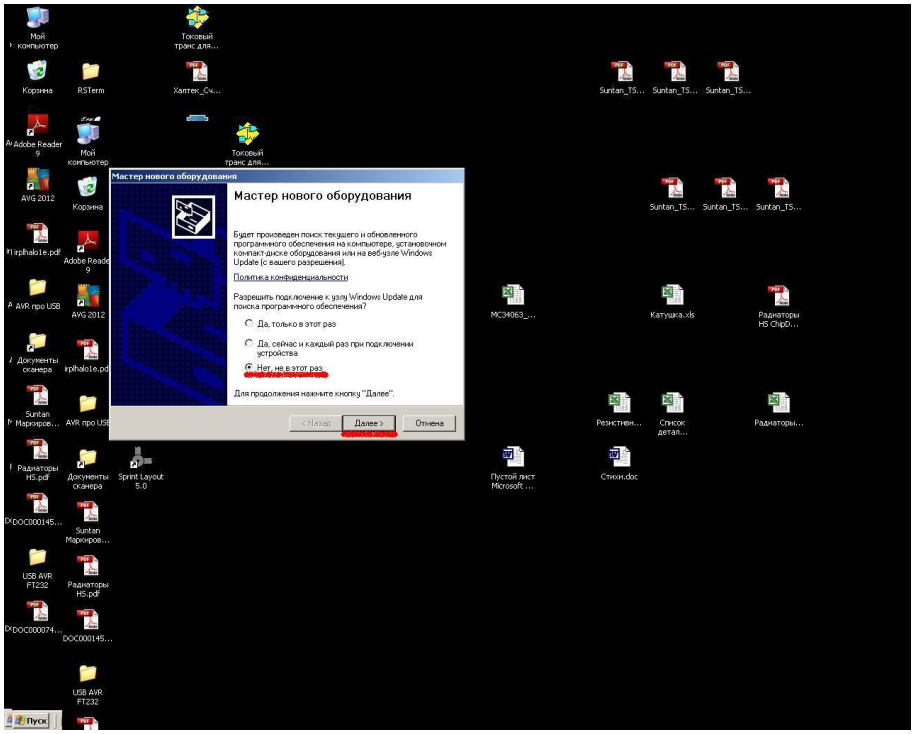


Рис. 12

Выбираем «Нет, не в этот раз» и нажимаем «Далее». На вопрос, откуда будут устанавливаться дрова для нового оборудования, отвечаем «Установка из указанного места» и жмем «Далее» (*Рис. 13*):

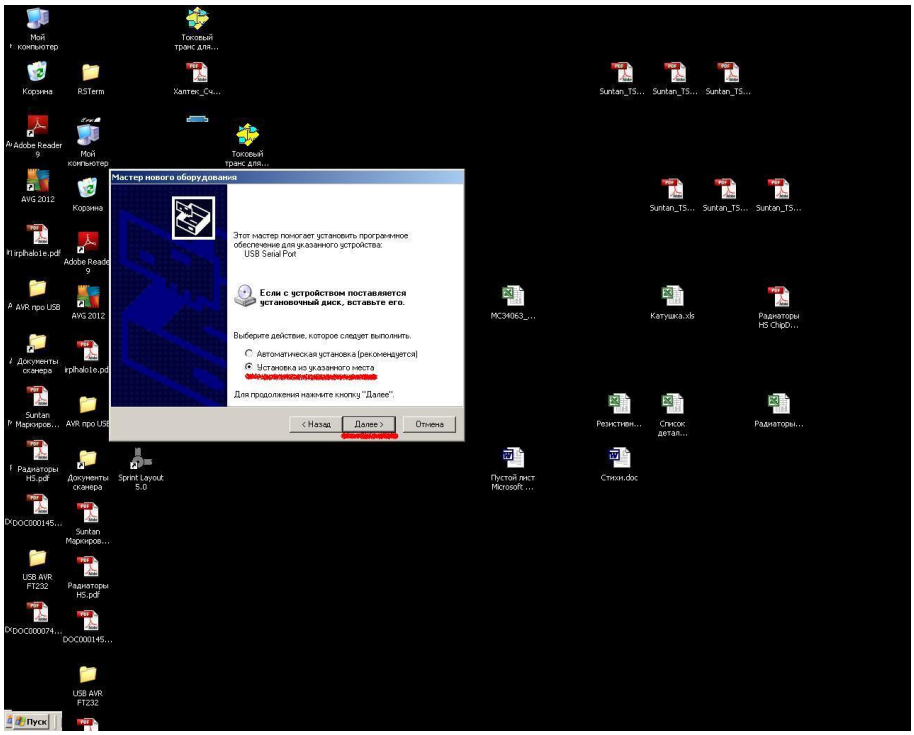


Рис. 13

Путь к местонахождению дровяных файлов должен сохраниться. Проверяем, и если это так, то жмем «Далее» (*Рис. 14*):

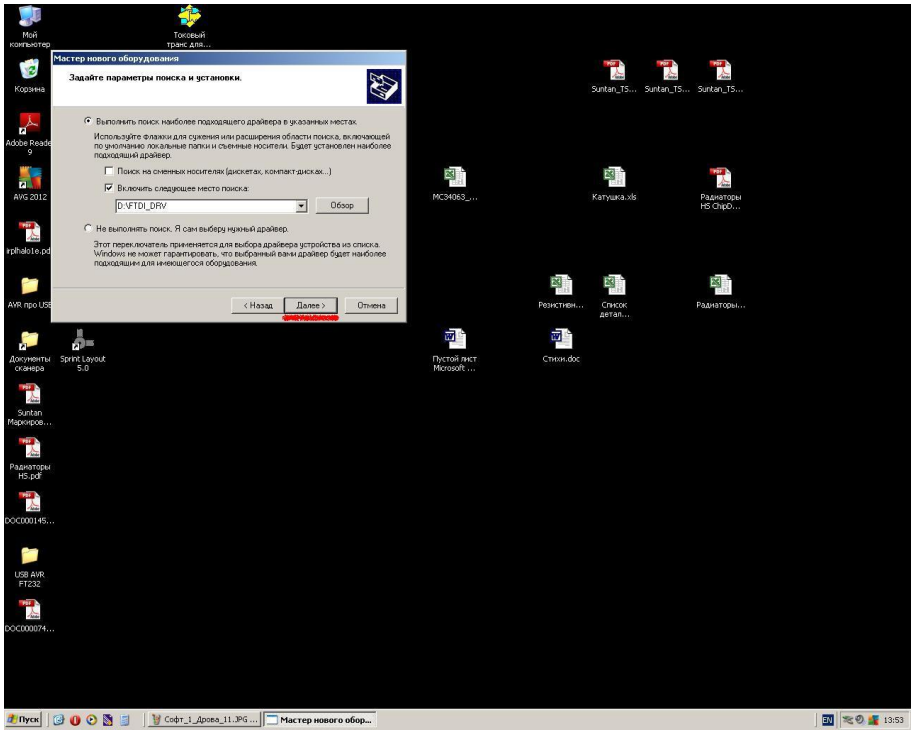
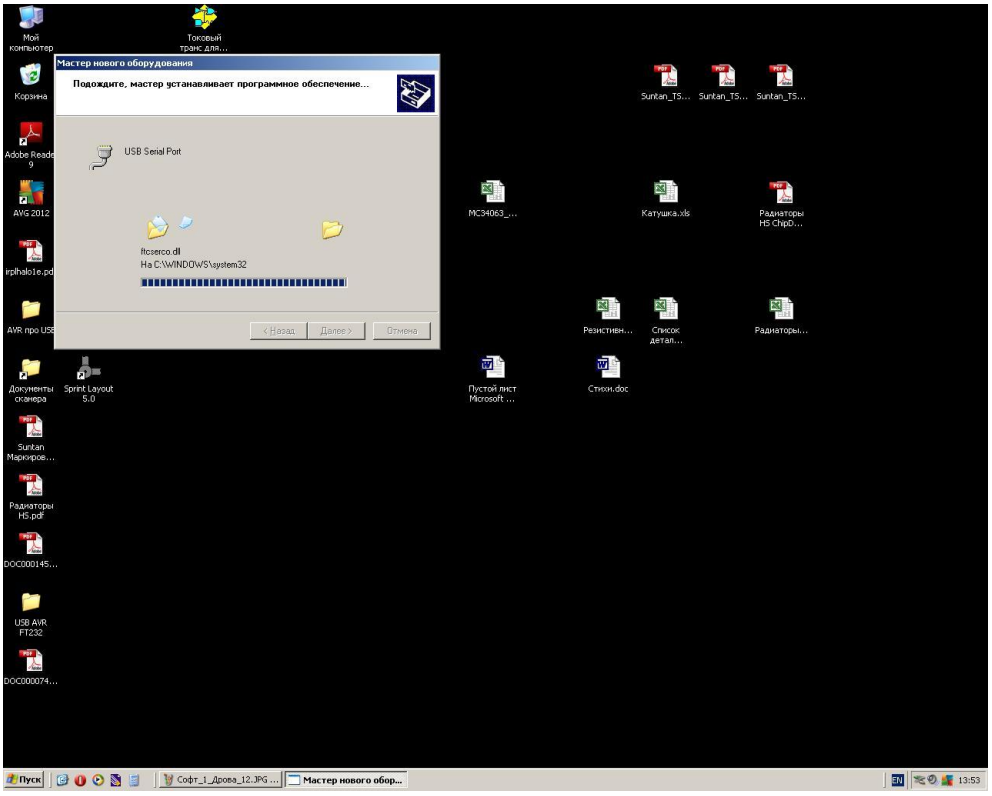


Рис. 14

Ждем, пока система найдет и установит дрова для последовательного порта USB, и нажимаем «Готово» (*Рис. 15*):



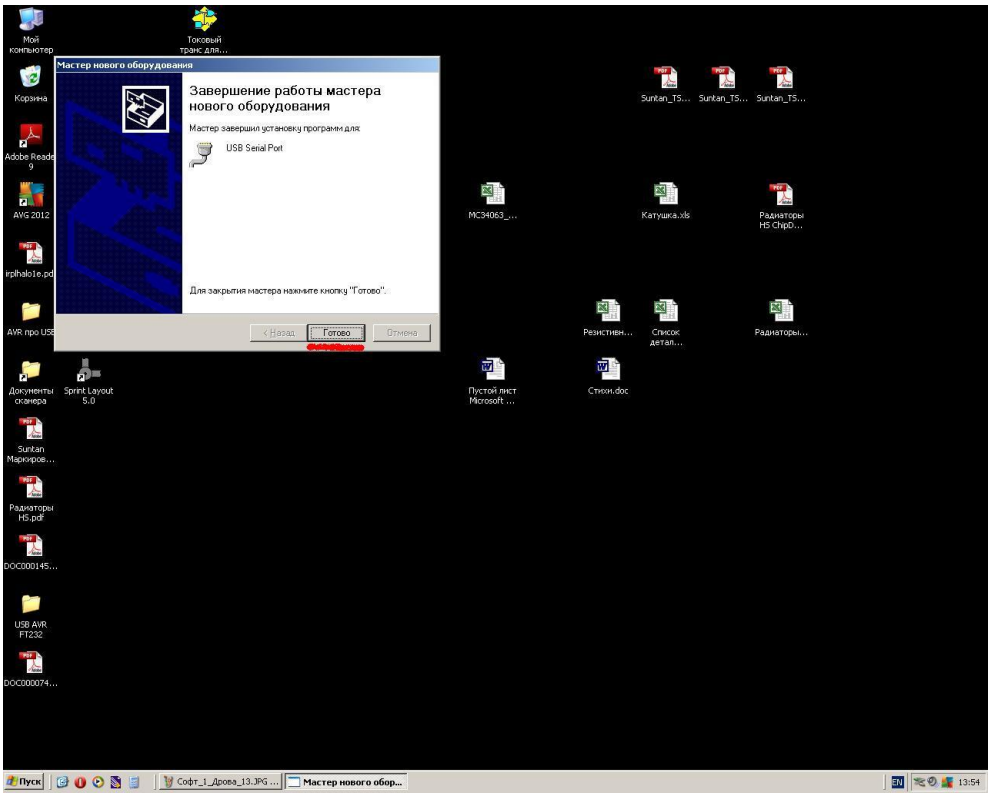


Рис. 15

Драйвера для программатора установлены. Для того, чтобы убедиться в этом, рекомендую снова зайти в диспетчер устройств и посмотреть обновленный список установленного

оборудования в разделах «Контроллеры универсальной последовательной шины USB» и «Порты (COM и LPT)» (*Рис. 16*):

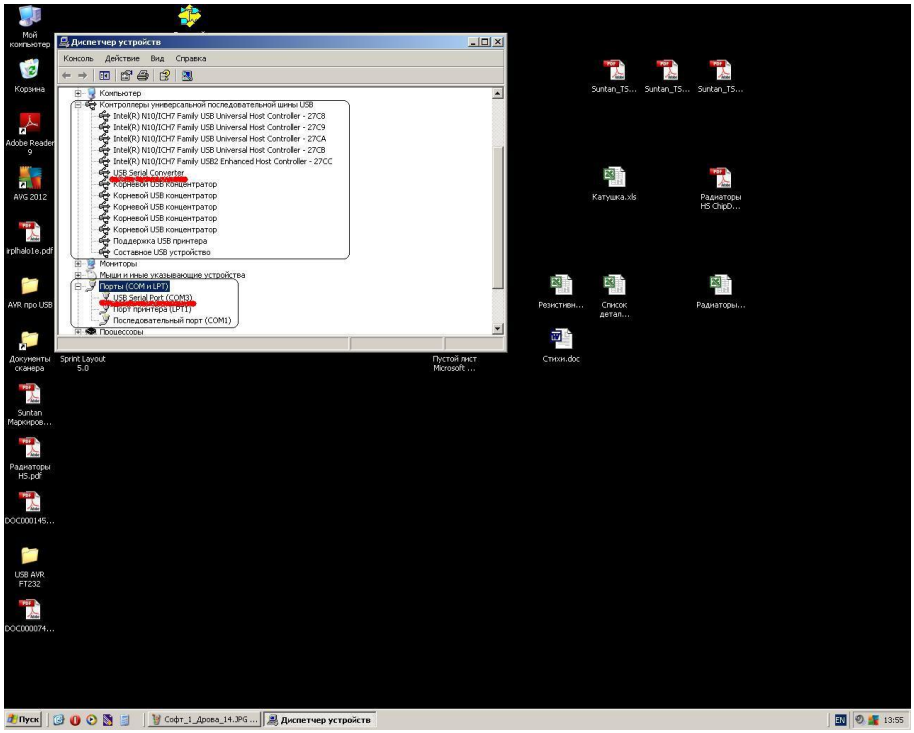


Рис. 16

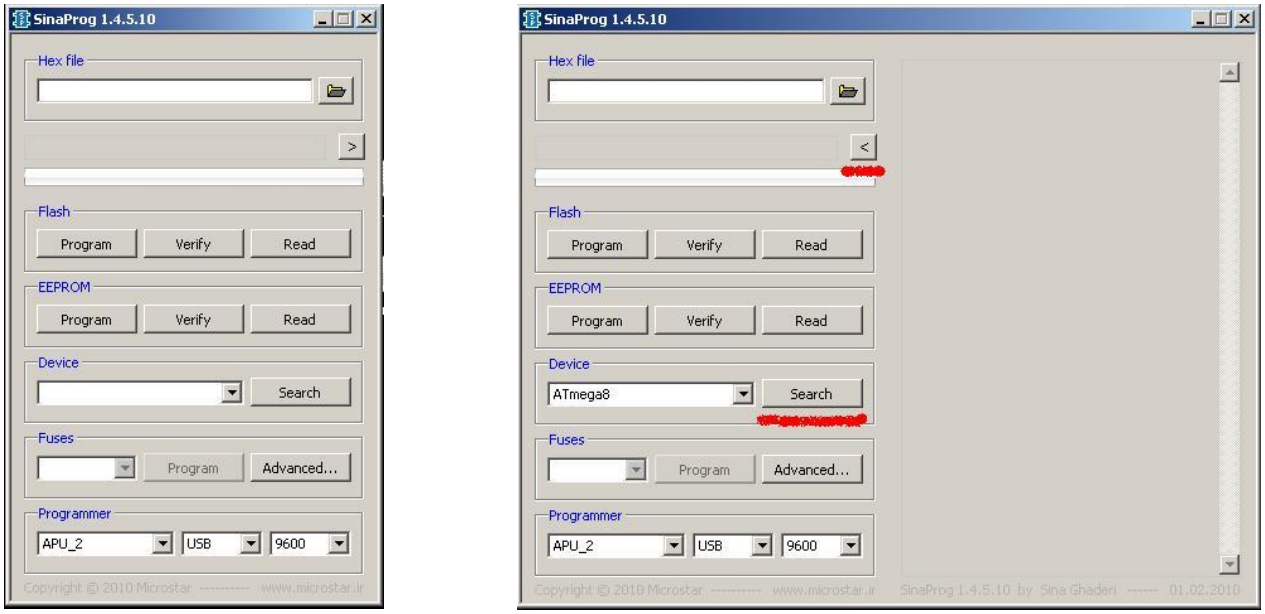
Как видно из *Рис. 16*, в системе появилось два новых устройства: новый последовательный порт (USB Serial Port (COM3)) и новый контроллер шины USB (USB Serial Converter).

Теперь осталось только запустить программу для прошивки микроконтроллеров. В качестве оной, как уже говорил выше, рекомендую использовать ПО SinaProg. Собственно, сама SinaProg является только графической оболочкой (однако, весьма и весьма удобной) к утилите AVRDUDE, которая и позволяет прошивать микроконтроллеры. Так вот, AVRDUDE – очень могучий инструмент. Поддерживает, наверное, все известные программаторы для камней AVR (в случае необходимости требуемый программатор добавить – дело пятнадцати минут). Перечень поддерживаемых микроконтроллеров также впечатляет (и, опять же, добавить в «библиотеку» новый кирпич не сложно). Если бы я о ней знал на момент ковыряния с AVR910 и USBasp, то, возможно, никогда бы не стал собирать программатор на основе FT232RL. Однако, звёзды выпали именно так, что о хорошем софте для работы с программаторами на базе чипов AVR я узнал только тогда, когда эти программаторы стали мне уже не нужны☺.

Итак, в качестве последнего шага запускаем программу SinaProg. Как говорилось ранее, архив с ней лучше распаковать прямо на диск C, чтобы путь к папке **SinaProg** был таким: *C:\SinaProg*, без всяких русских названий папок. Дважды щелкаем по файлу SinaProg.exe – и вот перед нами главное окно программы. В строке «Programmer» нужно выбрать (слева направо) «APU_2» или «USBBit» (с точки хрения микросхемы FT232RL это идентичные программаторы), далее «USB» и дальше «9600» (*Рис. 17а*).

«APU_2» или «USBBit» – это, как нетрудно догадаться, название используемого программатора. «USB» – порт, через который компьютер осуществляет связь с программатором. Если в компьютер воткнуто только одно устройство, содержащее микросхему FT232Rx (т.е. только программатор), можно смело выбирать порт «USB» и работать дальше. В случае же

подключения нескольких дывайсов с FT232Rx на борту требуемый номер порта для программатора иногда приходится определять методом перебора («USB», «USB1», «USB2», «USB3»). Поэтому при первом знакомстве с APU-2 для простоты лучше отключить все устройства USB, общающиеся с компом через FT232Rx, кроме самого программатора. Ну, а «9600» – это скорость программирования. Чтобы не углубляться в дебри, просто скажу очевидное – чем больше данное число, тем шустрее шьется камень☺. В большинстве случаев при длине шлейфа от программатора до прошиваемой платы не более 1м программа успешно заливается на максимальной скорости (3000000). Однако, бывают случаи когда для нормальной работы программатора скорость прошивания приходится снижать. Так что на первых порах можно ограничиться значением скорости «9600» (при такой скорости глюков не было обнаружено ни разу), а по мере знакомства с «характером» программатора при необходимости постепенно ее повышать.



а) *б)*

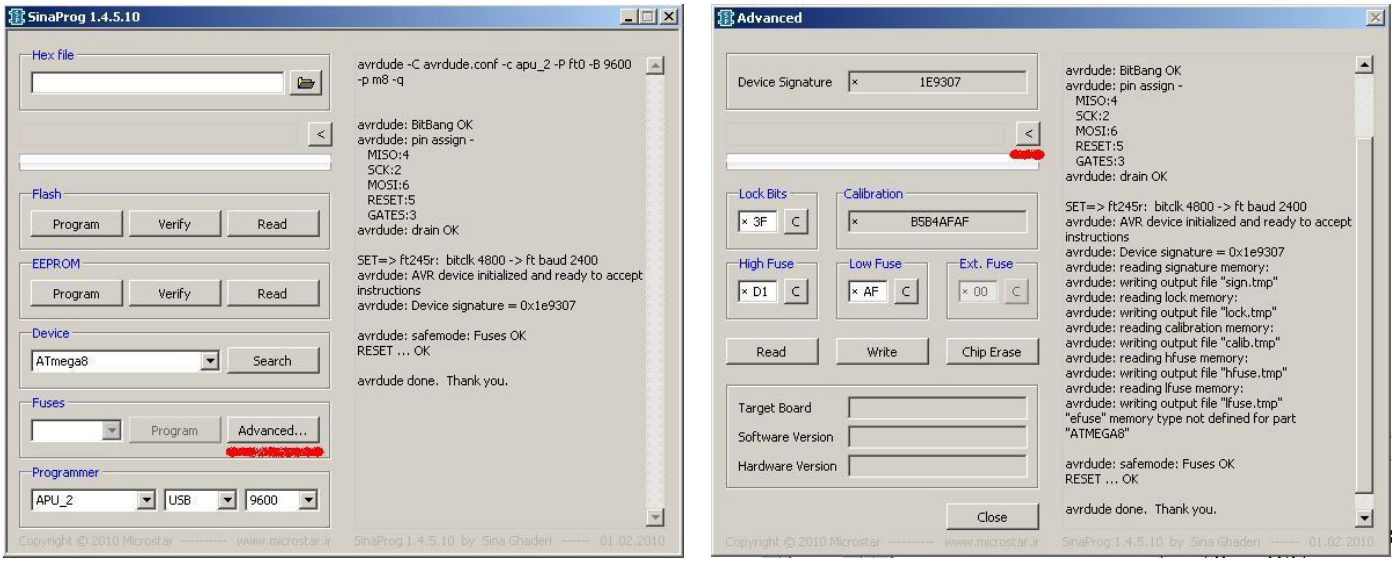
Рис. 17

Далее рекомендую нажать на символ « > » под значком «Папка» (*Рис. 17б*). Справа откроется окно, в котором в дальнейшем будет отображаться различная служебная информация о процессе работы программы SinaProg. На начальной стадии работы с данным софтом будет нелишним периодически поглядывать на появляющиеся сообщения. В принципе, при отсутствии желания разбираться с работой программы (что при первом знакомстве с неизвестным софтом вполне естественно) вникать в суть отображаемого не нужно, главное – чтобы не выскочило коварное слово «Failed», «Error» или «Invalid». Ну или еще что-то в этом роде. Вот тогда уже придется детально разбираться с возникшей проблемой при помощи служебной инфы. Отсутствие же коварных слов означает, что программа и программатор работают нормально, и причин для беспокойства нет.

Ну и последний шаг, требуемый для проверки работоспособности программатора – подключение его к плате, в которую установлен прошиваемый камень. Дальше выбираем в окне «Device» тип микроконтроллера, с которым будем работать (я для иллюстрации работы SinaProg взял плату с ATMega8A-AU на борту) и нажимаем кнопку «Search» (*Рис. 17б*).

Если программатор собран правильно, если его детали не издохли, если SinaProg и AVRDUDE не дуркуют, если программатор правильно подключен к прошиваемой плате, и вообще – если все прошло успешно, то в окне для служебной информации будет примерно то, что показано на *Рис. 18а*. В смысл написанного пока вникать не будем, обратим внимание лишь на отсутствие коварных слов «Failed», «Error» и «Invalid». И если они действительно отсутствуют, Вас можно поздравить: связка [комп] – [программатор] – [прошиваемая плата] работает успешно. О нормальной работе программатора также косвенно сигнализирует светодиод HL1 («PROG»). Напомню, что он должен загораться только на момент общения компа с прошиваемой платой (ну и, соответственно, с программатором).

Теперь можно шить нужные конфигурационные биты (ну, хотя бы для того, чтобы выставить нужный источник тактовой частоты для микроконтроллера). Для этого в строке «Fuses» нужно нажать кнопку «Advanced» (*Рис. 18а*).



а) ***б)***

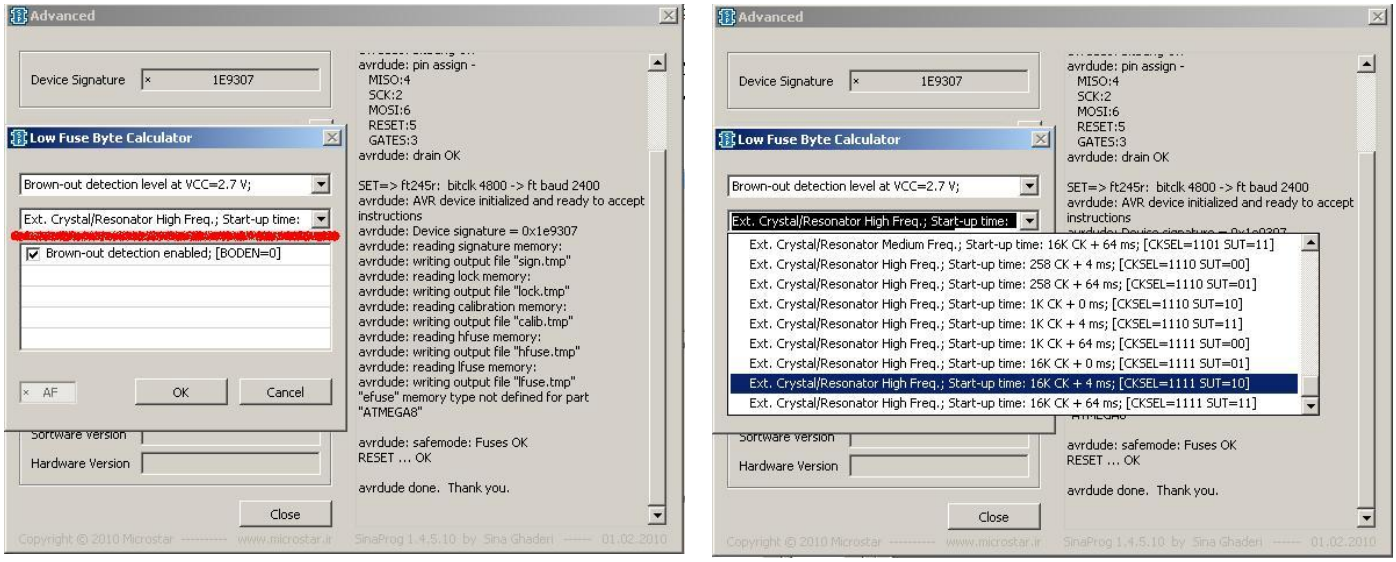
Рис. 18

При нажатии на данную кнопку SinaProg свяжется с прошиваемым камнем через программатор и считает с него установленные конфигурационные биты (фузы). В вывалившемся новом окне, как раз посвященным работе с фузами, также рекомендую нажать на символ « > » (*Рис. 18б*), просмотреть список сообщений справа и убедиться в отсутствии слов «Failed» или «Error». Ну а дальше можно уже выставлять нужные значения конфигурационных битов.

Здесь хочется сделать Важное Отступление. Желательно при работе с фузами выработать следующий рефлекс: перед какими-либо изменениями считывать состояние микроконтроллера кнопкой «Read» (на *Рис. 18б* находится под окном «High Fuse»). Этим Вы дополнительно обезопасите себя – ведь изменения будут вноситься в уже рабочий набор битов, а не какой-то левый (или, что еще ужаснее – «нулевой» или «единичный», когда все фузы сброшены или установлены соответственно). Левый набор может сформироваться множеством способов – например, от кратковременного глюка при считывании фузов с камня. В этом случае значения старшего и младшего байтов фузов, а также битов защиты в ряде случаев могут оказаться непредсказуемыми. И если мы не отслеживаем появление коварных слов в служебных сообщениях SinaProg, ковыряться мы будем в неадекватном поставленной задаче наборе конфигурационных битов. Как следствие – можно допрограммировать камень до такого состояния, когда его можно будет оживить только на параллельном программаторе или просто-

напросто выкинуть. А раньше (во времена тотального использования «лошади») были еще такие кадры – перед установкой фузов нажимали кнопку «Clear All» вместо «Read». Это, стало быть, мы установили состояние битов по умолчанию (ясно ведь даже дебилу, что состояние Default – это когда все «нули»). Естественно, после мастерской прошивки контроллера от данных экспертов неслись вопли: «Я запорол дорогущий микроконтроллер!!! Программа Pony Prog – говно, потому что совершенно непонятно, как в ней работать!!!». Так что, «аккуратнее, аккуратнее надо быть» с фузами.

Итак, нужные значения конфигурационных битов выставляются в общем случае в окнах «Lock Bits», «High Fuse», «Low Fuse» и «Ext. Fuse». Для установки нужных значений необходимо нажать на кнопку «C» («calculate», судя по названиям файлов в папке SinaProg) в соответствующем окне. Однако далеко не для всех моделей микроконтроллеров AVR будут доступны все возможные конфигурационные биты. Например, для ATMega8, как видно из *Рис. 18б*, окно «Ext. Fuse» наглухо заколочено в полном соответствии с даташытом на данный камень. Поэтому давать какие-то общие рекомендации по прошивке фузов для всех микроконтроллеров AVR, поддерживаемых SinaProg довольно сложно. Для каждой конкретной модели камня набор конфигурационных битов будет своим (для того, чтобы точно узнать – каким именно, полезно прочитать даташит на микроконтроллер). Исключение составляют фузы, отвечающие за источник тактовой частоты камня и за время старта (они, наверное, самые «популярные»). Данные фузы для всех поддерживаемых микроконтроллеров устанавливаются через окно «Low Fuse». После нажатия на кнопку «C» в этом окне, вывалится еще одно окошко, в котором, в частности, можно указать (*Рис. 19а*) нужные источник тактовой частоты и время старта из довольно нехилого списка, включающего в себя все возможные комбинации (*Рис. 19б*).



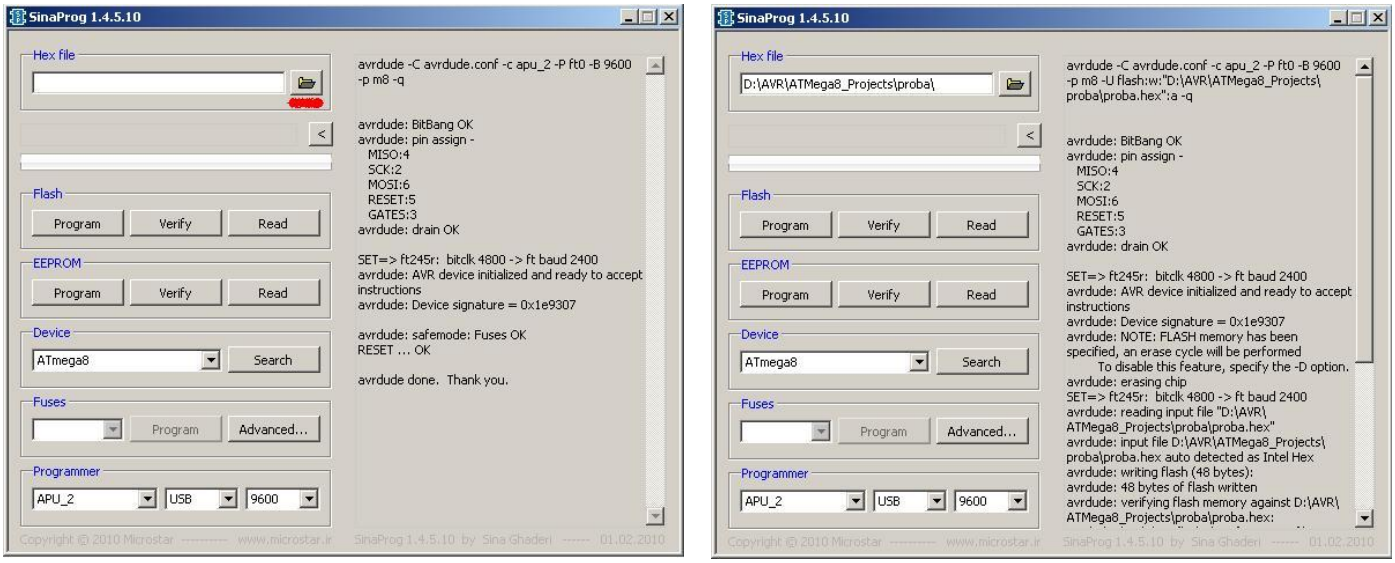
а) ***б)***

Рис. 19

Ну и после прошивки требуемого набора конфигурационных битов и битов защиты осталось только залить в камень исполняемую программу и (если надо) данные в EEPROM. Делается это просто. Переходим в главное окно программы (*Рис. 17*), нажав на кнопку «Close» в окне для работы с фузами (*Рис. 18б*). Далее нажимаем на кнопку «Папка» в строке «Hex File» (*Рис. 20а**Ошибка! Источник ссылки не найден.***). В проводнике выбираем нужный файл прошивки (*.hex для прошивки исполняемой программы во FLASH-память или *.eep для заливки данных в EEPROM). Насколько я понял, путь к файлу (и само имя файла) лучше не должны содержать русских букв, иначе иногда наблюдаются некие странные глюки – содержимое файла прошивки

может быть стёрто (но, возможно, это только у меня так). И завершающим этапом будет нажатие кнопки «Program» в соответствующей строке (для HEX-файла это «FLASH», для EEP-файла это «EEPROM»). Наблюдаем процесс заливки данных (при этом отмечаем скорость прошивки, решаем, следует ли ее увеличивать), после прошивки не забываем про коварные слова «Failed», «Error» и «Invalid» (*Рис. 20б*).

Ну вот и всё, можно радоваться исполнению программы микроконтроллером!



а) ***б)***

Рис. 20

Выводы

В качестве выводов хотелось бы рассмотреть плюсы и минусы программатора (естественно, это только мои личные ощущения).

Плюсы, на мой взгляд, таковы:

- отсутствие в программаторе элемента, требующего предварительной прошивки (исполняющего микроконтроллера) – особенно полезно для новичков;
- в большинстве случаев – высокая скорость прошивки микроконтроллеров AVR;
- возможность простого расширения функций программатора – использование APU-2 в качестве преобразователя USB-UART (присутствует по умолчанию), а также в качестве генератора прямоугольных импульсов с частотой 6/12/24/48МГц (см. даташит на FT232Rx); возможность использования линий CBUS0-CBUS3 как линий обычного порта ввода-вывода (см. даташит на FT232Rx); использование программатора в качестве источника питания +3,3В/50мА и/или +5,0В[1]/80мА (присутствует по умолчанию, но при определенной настройке APU-2 может по цепи +5,0В отдавать ток до 480мА).
- относительная дешевизна;
- простота схемотехники;
- простота печатной платы;

[1] Точнее, напряжения питания, снимаемого с шины USB

- малые габариты;
- возможность установки непосредственно в разъем USB компьютера, без использования дополнительных проводов (USB A-A, USB A-B);
- совместимость разъема XP2 («ISP») с широко используемым «стандартом» STK200/STK300;
- доступность фирменных драйверов на микросхему FT232RL для многих современных ОС;
- поддержка программатора мощнейшей программой-прошивальщиком AVRDUDE (и, как следствие, ее удобной графической оболочкой SinaProg);
- отсутствие необходимости дополнительной настройки программатора для выполнения им своих основных функций.

В качестве минусов программатора даже не знаю, что указать. Разве что, относительную сложность монтажа сам*о*й микросхемы FT232RL. Но по мне – так особо сложного здесь ничего нет, думаю, у Вас всё получится.

UPDATE: В Интернете люди часто жалуются на дороговизну микросхемы FT232RL. Граждане! Не смешите окружающих! Даже если не сравнивать по стоимости данный программатор с программаторами на основе чипов AVR, цена[1] за все детали в Интернет-магазине «Чип-НН» (именно в нем я закупаюсь) не превысит 200р. (150р. сама FT232RL, 8р. – буфер 74HC125D, ну и еще 40р. на всевозможные разъемы/резисторы/конденсаторы/светодиоды). Эта сумма для большинства – на один раз пива попить (даже не водки!). Причем – довольно средне попить. Ну а уж в сравнении с программаторами на базе микроконтроллеров AVR разница в цене составит максимум 100р (55р. – самый дешевый чип ATMega8A-AU, плюс кварц, плюс конденсаторы на кварц, плюс те же самые разъемы/резисторы/конденсаторы/светодиоды). Мне кажется, что хотя бы такая дополнительная функция FT232RL, как преобразователь USB-UART, влегкую отбивает эту разницу. Хотя, конечно, выбор за Вами, тут все зависит от конкретного случая (у того же меня есть в запасе пара десятков уж*е* приобретенных тех же ATMega8). Однако говорить о запредельной цене собранного программатора как-то, мягко говоря, странно.

В общем – желаю приятной работы с микроконтроллерами AVR и всех благ!

podkassetnik@yandex.ru

[1] На момент написания заметки

История изменений документа

Версия документа	*Дата*	*Изменение*	*Стра-ница*
1.0	02.10.12	Первая редакция документа	-