

## **Как работает системный загрузчик эмулятора AVR JTAG ICE или очевидное требует проверки.**

Автор Кротевич Виталий aka Vit.

[sensor@ugt.kiev.ua](mailto:sensor@ugt.kiev.ua)

Известная проблема при производстве клонов эмулятора от фирмы Атмел – это неполная программная совместимость в части загрузки новой версии прошивки. Складывается эта проблема из нескольких составляющих.

Во-первых, загрузчик фирменного прибора не позволяет прочитать память программ, в том числе себя.

Во-вторых, свежая прошивка поставляется в супер-формате \*.ebn, который конвертировать в общепринятый \*.hex толком нечем.

В-третьих, есть открытая информация, которой нет основания не доверять. Складывается она из двух не противоречащих друг другу частей – информация, которую показывает AVR Studio при работе с фирменным прибором, ну и информация от проекта BootICE.

Ни у кого не вызывает сомнений то, что загрузчик расположен в бутовой области, которая начинается с адреса 0x3800 в байтах или 0x1C00 в словах.

Итак, начнём. Начинаем доверять но проверять.

Супер-формат \*.ebn по логике вещей должна понимать та программа, которая выполняется при апгрейде фирменного прибора – оказывается это обычный AVRProg, который лежит в каком-то каталоге в районе инсталлированной AVR Studio. Вот только AVRProg может залить из файла \*.ebn прошивку, а вычитать её из камня загрузчик, ясное дело, не даёт. Вспоминаем тему и завязываем узелок – есть же открытые Application Notes AVR109 и AVR910, в которых описано то, что позволяет использовать AVRProg для заливки и вычитывания в отдельно стоящий камень. Т. е. надежда использовать построенный собственными руками программатор имени AVR910 есть – главное иметь нужный чип, хотя можно и любой другой с памятью программ поболее. Правда страшновато, т. к. неизвестно, пишутся ли фузы.

Идём на сайт проекта BootICE и добросовестно скачиваем всё, имеющее отношение к проекту. Находим файлы загрузчиков для ATmega163 и для ATmega16, выполняем всё, что там рядом написано, и приходим к выводу, что уже нужна прошивка не 4.07, а супер-свежая 4.08, ну и апгрейдить не получается за неимением файлов прошивки в формате \*.hex. Скверно. Ждём подарка от добрых самаритян.

Кроме того всё чаще кажется, что ну очень неинтересно вот так вот брать и каждый раз (сколько ж тех прошивок ещё будет!) мучать бит BootStart. Как-то это не по-эмбеддеровски, чтоли...

И вот он настал, о миг пленительного счастья, и нам выложили файл в котором, якобы, супер-свежая версия прошивки 4.08. Пробуем “проапгрейдиться” и, о страшновыговорить, узнаём, что загрузчик для ATmega163 не стирает память программ. (А как заливал прошивку 4.07, как дышал...) Можно было не ждать версий и просто попробовать стереть и прочитать чистый чип, но мы не ищем лёгких путей... и надеемся на халяву.

Но мы же в школе учили Бейсик, а в институте нам рассказывали о С. Опускаемся до чтения исходников загрузчиков от BootICE, написанных на Бейсике, дейташитов на ATmega163 и ATmega16, внимательного изучения Application Note AVR109. Далее пытаемся наваять свой загрузчик (не побоимся этого слова), чтобы “все боя-ались, чтоб не насмеха-а-ались”, в смысле чтобы и ежели чего не так подправить, ну и понимать как оно там шевелится.. Для начала пытаемся найти, что же за версия Бейсика была использована. Поиск выдал четыре известных природе варианта. Честно сливаем весь этот хлам, по дороге находим аспирина, ну и с сожалением узнаём, что в проекте BootICE использовано что-то особенное, неизвестное природе. Поправить код не удастся... Лёгкое знание С подталкивает к попытке развернуть код проекта AVR109 и найти 10 отличий с BootICE. Основные

отличия честно оформляем таблицей, переписанной из AVR109 и слегка дополненной. Ну очень похожие загрузчики оказываются. Последний вариант таблицы содержит определённые исправления и допущения. По образу и подобию BootICE калечим код AVR109, не забываем исправить файл \*.xcl под ATmega163, решаем сделать вход в программу пользователя через проверку в загрузчике одного из портов (точнее одного вывода), т. е. как в AVR109, только вывод выбираем другой – PC0, далее устраняем механические ошибки и... “включаем – не работает”. Точнее так же не умеет стирать, а с остальным – всё так же прилично. Вывод – ошибка в логике и, похоже, не в AVR109. Внимательно смотрим отличия по команде стирания – ещё раз наблюдаем резкое отличие – реакция на команду 'e' в BootICE соответствует нераспознанной команде, зато есть неизвестная команда 'U'. Так как мы ничего не теряем, объединяем реакцию на 'e' и на 'U' (страшно убирать реакцию на 'U', потому как тот кто писал, наверно, что-то знает...). Включаем – работает, т. е. стирает и шьёт. Море удовольствия. Мы гордимся нашей радостью за нашу гордость от нашей радости... (С).

И это был бы ...конец, но где же пистолет?(С) Почему Атмел может, а мы кто тогда? Почему у них загрузчик, размещающийся там же, где наш, работает, а наш нет?

По полезному совету крутого товарища выполняем самое сложное - отыскиваем на харде старинный дистрибутив AVR Studio 3.52. Инсталлируем это, например, на вторую ОС на том же компьютере, и находим прошивку в формате \*.hex. Запустив замечательный программный продукт по имени IDA и натравив его на прошивку из этой версии, получаем при беглом просмотре переход в бутовую область и, естественно, там ничего не лежит, потому как лежать должно, но только злые Плохиши знают, что там спрятано. А самое главное – мы теперь точно знаем, что переход ведёт не на начало бутовой области (адрес 0x3800), а дальше!!! - на адрес 0x3C00 в байтах (0x1F00 в словах). Этого не ожидал ни один клонирующий, даже Olimex. Правим в проекте файл \*.xcl, убираем вход в программу пользователя из загрузчика, зашиваем это дело вместе с прошивкой от AVR Studio 3.52 со стартом не из бутовой области, а с 0x0000. Включаем всё это с AVR Studio 4.07 и видим, что для апгрейда нужно подключить целевое устройство, выполняем непонятное требование и... всё получилось!!!

Для косметики размер бутовой области устанавливаем фузами с 0x3800 и ставим защиту от чтения бут-сектора. Защита от стирания приветствуется, но AVRProg (как-будто) не предлагает в неё писать по-любому (т.е. прошивки не пытаются заполнить эту область).

Дальше нас начинает беспокоить тревожное будущее – Атмел снял с производства ATmega163 и предлагает как замену ATmega16. Внимательнейшее прочтение AVR109 показало, что подпрограммы стирания и т.п. уже заточены под ATmega16, но немного нужно подрихтовать инициализацию портов и подправить код на предмет замены файла описаний. Но этого мало – у ATmega16 другие байты сигнатуры и другой идентификатор. Пытаемся разобраться. Сигнатура ATmega16 в BootICE неправильная и автор об этом прямо говорит, посему, доверяя дейташиту, вписываем сереньким в табличку и, соответственно, правим текст программы. С идентификатором всё оказалось хуже. Идентификатор (тип поддерживаемого у-ва) для ATmega163 не соответствует дейташиту и неясно, почему, но работает, посему доверимся BootICE и вводим оттуда неправильный (?) идентификатор для ATmega16. Компилируем, включаем, проверяем – РАБОТАЕТ!!!

Название команды	Принятая команда- Host Writes		Формат ответа - Host Reads	
	ID	Data	Data	
Вход в режим программирования- Enter Programming Mode	“P”			13d (\r')
Автоинкремент адреса - Auto Increment Address	“a”		dd 'Y'	
Установка адреса - Set Address	“A”	ah al		13d (\r')
Запись в память программ, младший байт - Write Program Memory, Low Byte	“c”	dd		13d (\r')
Запись в память программ, старший байт - Write Program Memory, High Byte	“C”	dd		13d (\r')
Выполнение записи страницы - Issue Page Write	“m”	dd		13d (\r')
Чтение битов блокировок - Read Lock Bits	“r”		dd	
Чтение памяти программ - Read Program Memory	“R”		dd (dd)	
Чтение памяти данных - Read Data Memory	“d”		dd	
Запись в память данных - Write Data Memory	“D”	dd		13d (\r')
Стирание всего кристалла - Chip Erase	“e”, “U”	dd		13d (\r')
<b><i>Запись битов блокировок - Write Lock Bits</i></b>	<b><i>“l”</i></b>	<b><i>dd</i></b>		<b><i>13d (\r')</i></b>
<b><i>Запись битов защиты - Write Fuse Bits</i></b>	<b><i>“f”</i></b>	<b><i>dd</i></b>		<b><i>13d (\r')</i></b>
Чтение битов защиты - Read Fuse Bits	“F”		dd	
Чтение старших битов защиты - Read High Fuse Bits	“N”		dd	
Остаться в режиме программирования- Leave Programming Mode	“L”	dd		13d (\r')
<b><i>Выбор типа микросхемы - Select Device Type</i></b>	<b><i>“T”</i></b>	<b><i>dd</i></b>		<b><i>13d (\r')</i></b>
Чтение байтов сигнатуры - Read Signature Bytes	“s”		3*dd <b>0x1E</b> <b>0x94</b> <b>0x02</b> /0x03/	
Возврат кодов поддерживаемых микросхем - Return Supported Device Codes	“t”		n*dd <b>0x66</b> /0x64/	00d
Возвращение идентификатора программы - Return Software Identifier	“S”		S[7] “AVR BOOT”	
Возврат версии программы - Return Software Version	“V”		dd dd “21”	
<b><i>Возврат версии устройства - Return Hardware Version</i></b>	<b><i>“v”</i></b>		<b><i>dd dd</i></b>	
Возврат версии программатора - Return Programmer Type	“p”		dd 'S'	
<b><i>Включение светодиода - Set LED</i></b>	<b><i>“x”</i></b>	<b><i>dd</i></b>		<b><i>13d (\r')</i></b>
<b><i>Выключение светодиода - Clear LED</i></b>	<b><i>“y”</i></b>	<b><i>dd</i></b>		<b><i>13d (\r')</i></b>
Неопределённая команда - Unknown commands				“?”
Очистка буфера обмена – Flushing receiver buffer	0x1b		-	-

В таблице жирным выделены отличия от открытого (соответствующего Applcaton Note AVR109 и AVR910) протокола и константы, байты сигнатуры и ID указаны для ATmega163, серым цветом отмечены байты сигнатуры и ID, соответствующие ATmega16, а перечёркнуты неиспользуемые команды, которые считаются неопределёнными.

